Universidad Carlos III de Madrid

Escuela Politécnica Superior

Ingeniería Técnica en Informática de Gestión

Proyecto Fin de Carrera

UTILIZACIÓN DE MÁQUINAS DE ESTADOS Y ÁRBOLES DE DECISIÓN PARA EL VIDEOJUEGO STARCRAFT

Autor: Carlos Morales Casas

Director: Moisés Martínez Muñoz

Co-director: Daniel Borrajo Millán

Agradecimientos

A mis padres, por el apoyo y la paciencia que han tenido estos años de carrera.

A mis hermanos, porque me han ayudado a hacer todos estos años más divertidos.

A Teresa, por confiar en mí y estar conmigo en todo momento.

Resumen

Desde que en 1972 Atari lanzase el PONG logrando que los videojuegos se volvieran populares entre el gran público la sofisticación de estos no se ha detenido hasta nuestros días en los que se ha convertido en una gran industria. La evolución de la tecnología y la búsqueda de nuevos retos para ofrecer a los jugadores han logrado que la variedad y complejidad de los videojuegos haya crecido rápidamente en los últimos años.

La Inteligencia Artificial (IA) es una parte esencial en el desarrollo de un videojuego ya que interviene en el comportamiento de todos los objetos que no controla directamente el jugador por lo que, a medida que los videojuegos se hacían más complejos, el estudio de la IA implementada en estos se ha vuelto cada vez más interesante, destacando los juegos de disparos en primera persona, que han llegado a utilizarse para entrenamiento militar [1]. Otro de los géneros más interesantes para el estudio de la inteligencia artificial en los videojuegos es el de la estrategia en tiempo real ya que en este tipo de juegos todos los jugadores intervienen al mismo tiempo, sin turnos, por lo que las decisiones se tienen que tomar rápido y teniendo en cuenta condiciones que pueden variar de un momento a otro. Dentro de este género, *StarCraft* es uno de los que mayor éxito ha alcanzado tanto por la crítica como por el nivel de ventas.

En este proyecto se presenta el desarrollo de un jugador automático para el juego de estrategia en tiempo real *StarCraft: Broodwar* desarrollado mediante la combinación de una máquina de estados y un conjunto de árboles de decisión que, recogiendo información del juego, permitan tomar decisiones con el objetivo de ganar la partida. Para el desarrollo de este proyecto se ha utilizado JNI-BWAPI, un *framework* de programación en el lenguaje Java que posibilita la implementación de nuestro jugador automático en el juego.

Abstract

Since 1972, when Atari published Pong, the first videogame to became popular, videogames sophistication has not stopped to our days while it has become a huge industry. The evolution of technology and the search for new challenges to offer to the players have made variety and complexity of video games to grow fast in recent years.

Artificial intelligence is an essential part in the development of a video game because it is involved in the behavior of all objects that the player does not directly control, so, as games became more complex, the study of artificial intelligence implemented at videogames has become increasingly interesting, highlighting First Person Shooter games, which have been used for military training [1]. Another of the most interesting genres for the study of artificial intelligence in videogames is Real Time Strategy (RTS) because in these games all the players play at the same time, without turns, so decisions must be taken quickly and considering conditions which vary from one moment to another. Within this genre, StarCraft is one of the most successful titles both from critics and in terms of sales.

This project presents the development of an automatic player to play the real-time strategy videogame StarCraft: Broodwar developed combining a state machine and a set of decision trees that, collecting information of the game, take decisions with the goal of win the game. For the development of this project has been used JNI-BWAPI, a framework implemented with the Java programming language that enables the implementation of our automatic player in the game.

Índice General

Índice de figuras

Índice de tablas

Capítulo 1: Introducción

Los videojuegos se han convertido en la principal industria de ocio audiovisual e interactivo, con una cuota de mercado muy superior a la del cine y la música [AEGI, 2015]. Entre los distintos géneros actuales uno de los que tiene mayor éxito es la estrategia en tiempo real o Real Time Strategy (RTS). Estos juegos requieren el uso de un pensamiento táctico y la planificación de acciones para alcanzar la victoria. Suelen caracterizarse por dar una gran libertad al jugador para diseñar su camino a seguir, sin pausa para la toma de decisiones. Se debe estar atento a distintos hechos que ocurren al mismo tiempo y dar órdenes rápidas para superar las dificultades.

1.1 Descripción del problema

La mayoría de los juegos incorporan algún tipo de IA. Los desarrolladores han usado durante años la IA para dar vida a personajes aparentemente inteligentes en innumerables juegos como *Pacman* (Namco, 1980), el primer juego en implementar una máquina de estados para sus enemigos fantasmas [2], *Goldeneye 007* (Rare, 1997), que demostró las mejoras en la jugabilidad que podía aportar una IA más compleja o *Halo* (Bungie, 2001), el cual incorpora enemigos con desarrollada IA gracias a complejos árboles de comportamiento (Figura 1).

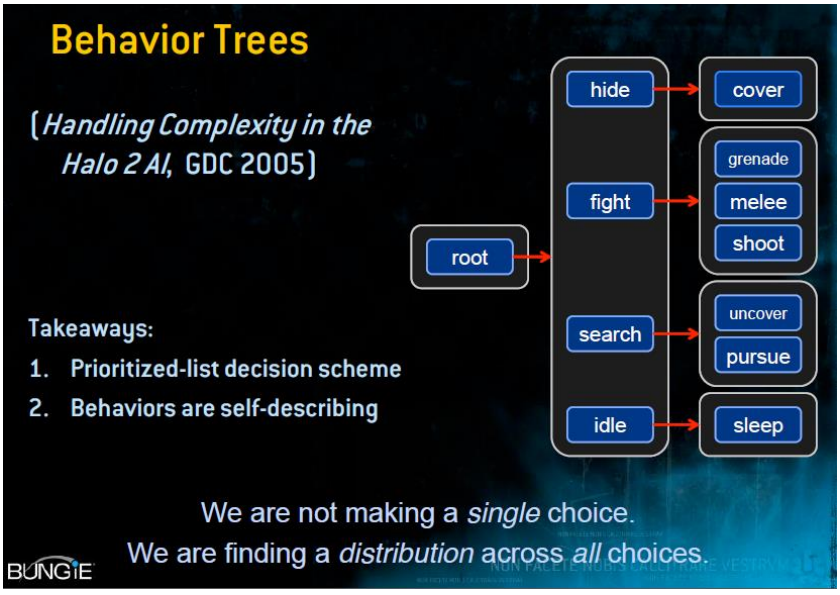

Figura 1. Esquema de un árbol de comportamiento de Halo 2

La IA es posiblemente una de las partes más difíciles del desarrollo de un videojuego ya que es responsable en gran parte de que el juego resulte entretenido o no. La adecuada implementación de la inteligencia del juego hará que el jugador se interese por un reto difícil pero sin desesperar por una dificultad demasiado elevada.

Al realizar el diseño de la IA a implementar es importante tener en cuenta los elementos que forman parte del juego. En un juego de estrategia en tiempo real es importante tener en cuenta tanto los recursos disponibles como las unidades, terreno o enemigos.

StarCraft, lanzado en 1998, es un videojuego perteneciente al género de la estrategia en tiempo real. Es uno de los videojuegos más vendidos de la historia con 9,5 millones de unidades vendidas en los primeros 10 años [Blizzard Entertainment, 2008] y uno de los más jugados en red. Con el paso de los años se ha convertido en uno de los juegos favoritos para desarrollar jugadores automáticos o robots (*bots*) con los que implementar distintas técnicas de IA para el control automático del juego llegando a celebrarse campeonatos en los que se compite por la IA capaz de conseguir mayor número de victorias.

Este proyecto comprende el desarrollo de un *bot* para el videojuego StarCraft: Broodwar que sea capaz de interactuar con el juego, analizar la situación del jugador y del enemigo en la partida y vencerle aprovechando las ventajas de las que dispone. Para lograrlo se debe de diseñar un sistema que gestione eficazmente los recursos disponibles en el juego, construir los edificios necesarios para poder entrenar soldados o fabricar vehículos, defenderse de posibles ataques enemigos y, finalmente, organizar ataques con los que se debilite al enemigo hasta llegar a la victoria.

1.2 Motivación

En primer lugar, elegí un proyecto relacionado con la IA porque, de todas las asignaturas cursadas durante mis estudios, la de IA es una de las asignaturas que me resultó más interesante, tanto teóricamente como en las prácticas realizadas. Por otro lado, los videojuegos fueron mi primer acercamiento con el mundo de la informática y es una afición que sigo manteniendo.

Durante el resto de la carrera los videojuegos han sido una materia que se ha tratado muy vagamente por lo que este proyecto supone, al mismo tiempo, la culminación del trabajo

desarrollado a lo largo de la carrera y mi primer acercamiento a los videojuegos como desarrollador aficionado, lo cual fue una de las causas de mi interés por la informática y del inicio de los estudios en la carrera. En un futuro, este proyecto podría convertirse en un inicio como desarrollador de videojuegos a nivel aficionado.

1.3 Objetivos del trabajo

El objetivo principal de este proyecto es el de realizar un jugador automático con el que jugar al videojuego *StarCraft: Broodwar* mediante la utilización de una combinación de máquinas de estado y árboles de decisión. Este objetivo se descompone en los siguientes:

1- Análisis del videojuego *StarCraft: Broodwar*. El primer paso para la realización del proyecto requiere conocer el funcionamiento del juego y su sistema de control. Para esto es necesario estudiar las distintas unidades de cada una de las razas, los objetivos del juego (como ganar partidas) y las estrategias a seguir para jugar con éxito.

2- Estudio de las técnicas más adecuadas para implementar IA en un videojuego de estrategia en tiempo real. Antes de la realización de un autómata es necesario conocer previamente cuales son las mejores alternativas para su diseño por lo que es necesario el estudio de las distintas técnicas existentes con las que implementar la IA con la que se realizará la toma de decisiones.

3- Definición del sistema de toma de decisiones. Una vez elegida la implementación del autómata a través de la combinación de máquinas de estado y árboles de decisión, una de las tareas más importantes que restan en el proyecto es la de definir los estados que compondrán el autómata, las acciones que estarán incluidas en cada estado y las transiciones de uno a otro así como el diseño de los árboles de decisión donde se ejecutan dichas acciones.

4- Análisis y estudio de la API. Para implementar correctamente el autómata diseñado es necesario estudiar el *framework* con el que interactuaremos con el juego, en este caso JNI-BWAPI.

5- Comprobación del correcto funcionamiento del jugador automático desarrollado. Después de implementar el jugador automático en el sistema se deben diseñar y ejecutar una serie de pruebas para verificar el correcto funcionamiento del jugador y si su

comportamiento se ajusta a los objetivos del proyecto y realizar las correcciones necesarias.

6- Desarrollo de la documentación del proyecto. Una vez finalizado el desarrollo del jugador automático se debe de realizar una memoria donde se plasme, de forma clara, los objetivos, el desarrollo y el resultado final del proyecto.

1.4 Estructura del documento

El documento se divide en los siguientes capítulos:

- En el primer capítulo incluye una breve introducción al proyecto, las motivaciones que han llevado al autor a su realización, se describen los objetivos marcados en el trabajo y, finalmente, un breve resumen de cada uno de los capítulos del documento.

- El segundo capítulo contiene una descripción de las áreas, técnicas y tecnologías utilizadas durante el proyecto. En primer lugar se analiza el mundo de los videojuegos para después profundizar de forma detallada en los juegos de estrategia en tiempo real y centrarse finalmente en el videojuego *StarCraft*, con el que se realiza este proyecto, describiendo los distintos tipos de razas y unidades, los recursos y la mecánica del juego. Posteriormente se realiza una introducción a la Inteligencia Artificial y su implementación en los videojuegos actuales y finalmente se hace una descripción de JNI-BWAPI, el *framework* con el que se desarrollan el jugador automático del proyecto.

- En el tercer capítulo se realiza una descripción detallada del proceso de análisis y diseño de la combinación de máquina de estados y árboles de decisión con la que se ha implementado el jugador automático. Para ello se realiza una descripción de las características funcionales y restricciones del sistema, los actores y casos de uso.

- El cuarto capítulo presenta la experimentación realizada para evaluar el funcionamiento del sistema, incluyendo una descripción detallada de las pruebas realizadas y de los resultados obtenidos.

- En el quinto capítulo describe el ciclo de vida elegido para el desarrollo del proyecto detallando todas las fases del proyecto, la planificación del mismo y sus costes descritos en un presupuesto que incluye las cantidades invertidas tanto en materiales como en personal.

- En el sexto capítulo se presentan las conclusiones obtenidas tras la realización del proyecto detallando las conclusiones para cada uno de los objetivos del proyecto y las posibilidades de continuar con este proyecto.

- Finalmente se incluye un conjunto de anexos. Dentro de estos se incluye el proceso de instalación del sistema y el manual de usuario para que el proyecto pueda ser utilizado por una persona sin conocimientos en el mismo. También se incorporan un conjunto de definiciones y las referencias utilizadas en este documento.

Capítulo 2: Estado del Arte

Según la Real Academia Española, un videojuego es “Un dispositivo electrónico que permite, mediante mandos apropiados, simular juegos en las pantallas de un televisor o de un ordenador” [3]. De forma más precisa se podría definir a los videojuegos como una aplicación interactiva realizada con el propósito de entretener que se ejecuta en un sistema informático provisto de una pantalla.

Aunque hace unos años era visto como un juguete más e incluso compañías jugueteras como Mattel llegaron a lanzar sus propias consolas de videojuegos [4], los videojuegos están estrechamente relacionados con el desarrollo de las computadoras de propósito general, las cuales surgieron en los últimos años de la Segunda Guerra Mundial, y han ido evolucionando de forma paralela llegando a ser una de las industrias del entretenimiento más importantes de la actualidad.

2.1 Evolución de los videojuegos

Los primeros antecedentes de los videojuegos tal como los conocemos en la actualidad se trataban en realidad de pruebas académicas o experimentos científicos. El que está considerado como primer precedente de los videojuegos actuales es el “Dispositivo de Entretenimiento de Tubos de Rayos Catódicos” (Figura 2) patentado el 25 de Enero de 1947 por Thomas T. Goldsmith y Estle Ray Mann [4]. Este dispositivo era un simulador interactivo de lanzamiento de misiles en el que se debía ajustar la curva y la velocidad de un punto en una pantalla CRT que simulaba ser un misil con la misión de impactar en unos objetivos, como por ejemplo un avión, colocados encima de la pantalla. Al no estar integrado el objetivo en el simulador era el propio jugador el que debía de determinar si el misil impactaba en el objetivo o no. A pesar de que su mecánica lo acerca a lo que sería un videojuego, la falta de gráficos impide considerarlo como tal.

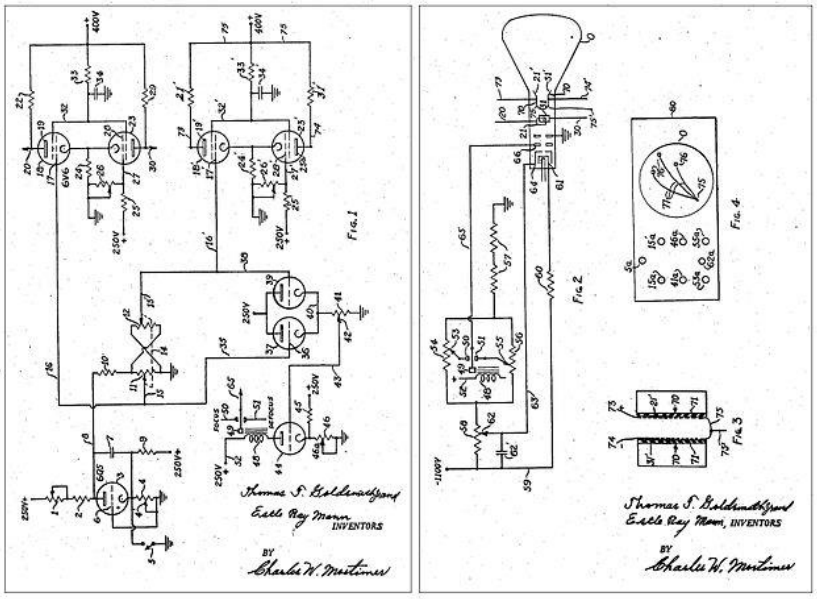

Figura 2. Patente del Dispositivo de Entretenimiento de Tubos de Rayos Catódicos

En marzo del 1950, Claude Shannon presentó junto al economista y matemático D. G. Champernowne un artículo llamado *Programming a Computer for Playing Chess* en la revista Philosophical Magazine [6]. En dicho artículo se describen técnicas y algoritmos necesarios para que un ordenador sea capaz de jugar al ajedrez de forma razonable, pero en ese momento no había ningún ordenador con el suficiente potencial para poder ejecutarlo, así que no fue hasta el noviembre de 1951 que el Dr. Dietrich Prinz pudo escribir el programa original en una Ferranti Mark I, la primera computadora electrónica comercial [7].

En 1952, Alexander "Sandy" Shafto Douglas presenta su tesis de doctorado en matemáticas en la Universidad de Cambridge (Inglaterra) basada en la interactividad entre seres humanos y computadoras y en la que incluye, para demostrar sus teorías, una versión computarizada del juego de las tres en raya llamada OXO, un programa que permitía a un humano jugar contra la máquina EDSAC (*Electronic Delay Storage Automatic Calculator*), la segunda computadora con la capacidad de almacenar programas electrónicos (siendo la primera la SSEM de la Universidad de Manchester) [8]. Es el primer juego de computadora que utiliza gráficos por lo que para muchos es considerado como el primer videojuego de la historia.

El 18 de Octubre de 1958 William Higinbotham, un físico nuclear estadounidense del Laboratorio Nacional de Brookhaven, presentó, durante un día de puertas abiertas, un programa que simulaba un partido de tenis llamado *Tennis for two* (Figura 3). El juego permitía a dos jugadores, a través

de sus respectivos controles, realizar saques y voleas de una pelota de tenis representada con un punto en la pantalla de un osciloscopio. Se crearon largas colas para probar el juego el día de su presentación. Existen algunas opiniones que consideran a este videojuego como el primero de la historia ya que *Tennis for two*, a diferencia de OXO, sí que emplea movimientos y tiene gráficos dentro del sistema.

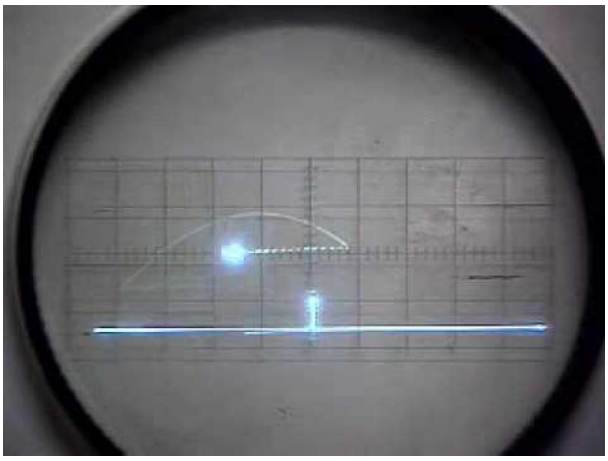

Figura 3. Imagen de Tennis for two

A pesar del debate que pueda existir entre los dos juegos anteriores, el juego que más influenció en el nacimiento de los videojuegos actuales fue *Spacewar!* (Figura 4). Diseñado por Steve Russell, Martin Graetz y Wayne Wiitanen siendo estudiantes del MIT en 1961, el juego permitía que dos jugadores manejasen su propia nave con la misión de destruir al contrario e implementaba campos gravitacionales así como otros efectos físicos. Estaba desarrollado usando el ordenador DEC PDP-1, lo que posibilitó añadir incluso un joystick para jugar [9]. Su éxito hizo que se implementaran versiones del juego en otras universidades y para ordenadores distintos e influenció en gran medida a los juegos posteriores. Incluso en 1971 Bill Pitts y Hugh Tuck desarrollaron una versión de *Spacewar!* llamada *Galaxy Game* que se convirtió en el primer juego en el que era necesario introducir monedas para jugar y alcanzando mucho éxito entre los estudiantes de la universidad de Stanford [10].

Figura 4. Monitor con el juego Spacewar!

Solo dos meses después de Galaxy Game, Nolan Bushnell y Ted Dabney lanzaron Computer Space. Este juego, aunque también se trataba de una versión de *Spacewar!* fue el primero creado con fines comerciales ya que se diseñó para instalarlo en máquinas recreativas con el propósito de venderlo a bares y otros lugares públicos. El juego no alcanzó el éxito esperado debido a su complejidad pero después de este fracaso Bushnell y Dabney crearon su propia compañía, Atari.

En abril de 1972, después de un largo desarrollo por parte de Ralph Baer, se presentó a la prensa la primera consola de videojuegos de la historia, la Magnavox Odissey [11]. Fue lanzada al público en agosto de ese mismo año con varios juegos que simulaban partidos de hockey, tenis o voleibol de forma muy sencilla ya que, por ejemplo, no disponían ni de marcador para la puntuación del jugador ni de sonido.

Sin embargo, no fue hasta noviembre de 1972 con el éxito de PONG (Figura 4), creado por Allan Alcorn para Atari, cuando comenzó su popularización a través de la proliferación de salones recreativos y las consolas de videojuegos domésticas. Este juego simula una partida de tenis de mesa y en él pueden participar dos jugadores uno contra otro o un jugador contra la máquina. Atari fue demandada por Magnavox al considerar que el PONG estaba basado en el juego de tenis incluido en su consola, pero eso no impidió el éxito del juego, lo que por otro lado ayudo a elevar las ventas de la propia Magnavox [12].

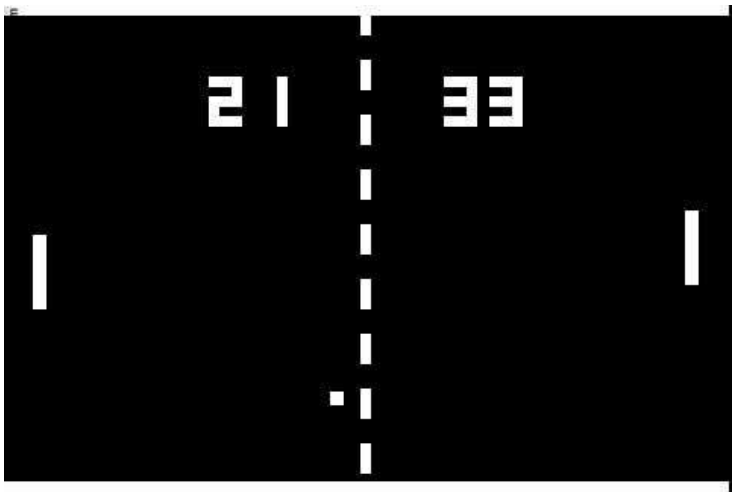

Figura 5. Captura de PONG

Después del enorme éxito alcanzado con PONG, Atari y otros desarrolladores se lanzaron a crear juegos con temáticas diferentes creando nuevos géneros. Así se empezó a crear una verdadera industria en torno a los videojuegos que pronto traspasó las fronteras de Estados unidos hasta llegar hasta Japón, país de las mayores desarrolladoras en los siguientes años ayudadas por el éxito de *Space Invaders* (Figura 6). Creado por Toshihiro Nishikado y lanzado por Taito en 1978, este juego fue uno de los primeros *shoot 'em up* o "matamarcianos" y consistía en manejar una nave espacial disparando a tus enemigos para conseguir la mayor cantidad de puntos.

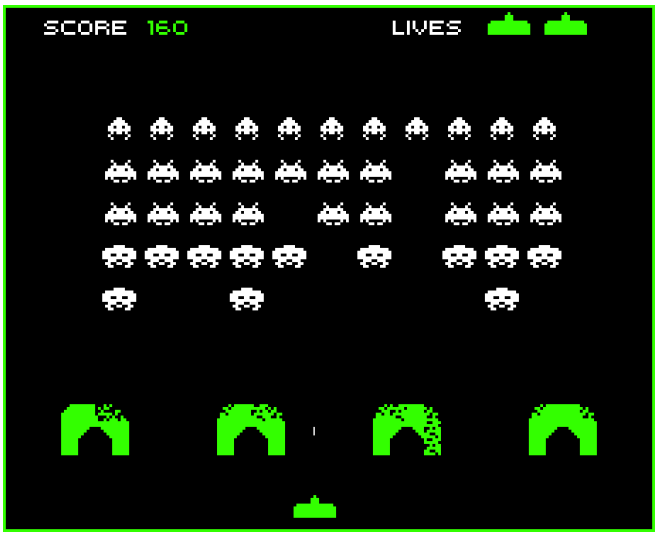

Figura 6. Captura de Space Invaders

Japón sería a partir de entonces el origen de las mayores compañías de videojuegos. En 1979 Namco lanzaba *Galaxian*, un videojuego de temática similar a *Space Invaders* pero que conseguía dar un paso más allá en los gráficos con mejores animaciones y más colores en pantalla. Nintendo incluso consiguió un casi monopolio en Estados Unidos con el lanzamiento de su consola *Nintendo Entertainment System* en 1985.

También a finales de los años 70 del siglo XX surgieron para los limitados ordenadores personales de entonces los primeros videojuegos del género de aventuras, distanciándose de la acción de los juegos más mediáticos de entonces. Los juegos de este género se caracterizan por la exploración, la interacción con el entorno y la resolución de problemas o puzles. Los primeros juegos de este género no disponían de gráficos, como *Colossal Cave Adventure* (1976), se limitaban a textos en los que se podía decidir de qué manera avanzar en la historia. Posteriormente el género fue evolucionando al tiempo que lo hacía la potencia de los ordenadores domésticos y en 1980 Roberta y Ken Williams, fundadores de Sierra Entertainment, lanzaron *Mystery House (Figura 7)*, el primer juego de aventura en hacer uso de gráficos [13].

Figura 7. Captura de Mistery House

A medida que la potencia de las máquinas aumentaba, se desarrollaron juegos con gráficos cada vez más elaborados y surgieron distintos subgéneros como *The Legend of Zelda*, lanzado por Nintendo en 1986 y que combina la acción con la aventura o *Maniac Mansion*, lanzado por Lucas Arts en 1987 revolucionando las aventuras gráficas ya que, gracias a su motor SCUMM (siglas de *Script Creation Utility for Maniac Mansion*), permitía jugar empleando solo el ratón, sin introducir comandos por teclado. SCUMM fue utilizado y mejorado en juegos de Lucas Arts posteriores como *The Secrect of Monkey Island* (Figura 8), creado en 1990 y considerado uno de los mejores juegos de todos los tiempos.

Figura 8. Captura de The Secret of Monkey Island

Durante los años 80 se desarrollaron la mayor parte de los géneros actuales. Los juegos de estrategia real, como se detalla en el siguiente punto del capítulo, tienen su origen en *Herzog Zwei* de Tecnosoft, lanzado en 1989. *StarCraft,* el juego con el que se desarrolla este proyecto, pertenece a este género. También en 1989 se lanza *SimCity* de Will Wright, siendo el primer juego de éxito en el subgénero de los simuladores de gestión, el más popular dentro del género de simuladores. Dentro de los juegos de acción se crearon distintos subgéneros como los juegos de lucha (siendo *Street Fighter II* (Figura 9) de Capcom el videojuego con más influyente dentro de los juegos de peleas) o de plataformas con éxitos como *Super Mario Bros.* (1985) de Nintendo o *Sonic* (1990) de Sega.

Figura 9. Captura de Street Fighter II

En la actualidad los videojuegos que gozan de mayor popularidad son los juegos de acción *sandbox*, traducible como "no lineal" o de "mundo abierto", un subgénero de los juegos de acción que permite cierta libertad al jugador a la hora de interactuar con el entorno. La serie *Grand Theft Auto*, con más de 800 millones de dólares recaudados el primer día de venta de su último juego [14] es la de mayor impacto dentro de los videojuegos de este tipo. Otros géneros muy populares en la actualidad son el deportivo, con franquicias como FIFA o NBA 2K, los de disparos en primeros persona con ejemplos como la serie *Call of Duty* o el propio género de la estrategia en tiempo real.

2.2 Juegos de Estrategia en Tiempo Real

Los videojuegos de estrategia en tiempo real o RTS (*real-time strategy* en inglés) son videojuegos de estrategia en los que no hay turnos sino que el tiempo transcurre de forma continua para todos los jugadores. Suelen estar basados en juegos de guerra.

En un RTS, los participantes posicionan y gestionan las unidades bajo su control para controlar áreas del mapa o destruir los bienes de sus oponentes. Generalmente en los RTS es posible crear unidades y estructuras adicionales durante el transcurso del juego con la limitación de los recursos disponibles en el mapa. Los juegos de estrategia en tiempo real se suelen caracterizar además por la posibilidad de investigar nuevas tecnologías para aplicar mejoras a las unidades bajo control del jugador así como por poder controlar indirectamente las unidades.

Herzog Zwei (Figura 10), lanzado en 1989 para la consola Sega Mega Drive, es considerado el primer videojuego en encajar con la definición de un juego de estrategia en tiempo real moderno [15] ya que su objetivo es superar al enemigo y destruir su base para lo que es necesario fabricar distintos tipos de unidades a las que después se les puede dar órdenes como patrullar la base o atacar. Cada jugador tiene una base principal, varias bases menores bajo su control, y algunas bases neutrales no reclamadas por ningún bando. La única unidad que se controla directamente es tu soldado, que puede cambiar de un avión a una unidad terrestre y el resto de unidades son únicamente de apoyo.

Figura 10. Imagen de Herzog Zwei

El concepto de controlar unidades individuales dándoles órdenes en tiempo real para ejecutarlas posteriormente no se había visto a ese nivel en un juego anteriormente. Las diferentes tropas y vehículos cuestan dinero y, además, es necesario gestionar el combustible del que se dispone, lo que sitúa a este juego como el precursor de la gestión de recursos, característica destacada de los RTS. A diferencia de la mayoría de los RTS posteriores no se puede reparar el daño a la propia base. Incluye un modo de pantalla dividida para que se pueda ver lo que el oponente (humano o informático) está haciendo. *Herzog Zwei* precedió al primer juego de estrategia en tiempo real de PC, *Dune II* de Westwood Studios, en casi tres años y, a pesar de eso, los ordenadores personales se volvieron pronto en la plataforma destacada del género.

A pesar de ser una continuación, el videojuego *Dune II* (Figura 11) es un juego que no tiene nada que ver con su predecesor, clasificado dentro del género de las aventuras gráficas aunque contiene algo de estrategia. La continuación supone una evolución del género con respecto a *Herzog Zwei* en varios aspectos y estableció las convenciones del género de estrategia en tiempo real. El juego incluye mejoras como poder construir bases en cualquier lugar del mapa y hacerlo en el orden que se elija. Además, el juego introduce los árboles de dependencia tecnológica [16]. Por ejemplo, para producir cosechadoras y tanques hay que construir una fábrica de maquinaria pesada y para ello se necesita primero tener una fábrica de maquinaria ligera.

Dune II también introduce la idea de que los oponentes o “Casas” pueden tener diferentes unidades y armas. Así, la Casa Atreides, Casa Ordos y Casa Harkonnen de *Dune II* tienen sus

propias "armas de la casa" especiales además de las estándar, lo que representa una gran novedad pero es un juego relativamente sencillo al compararse con los RTS modernos ya que tiene limitaciones como un número máximo de 25 unidades controlables.

Figura 11. Imagen de Dune II

Después del lanzamiento de *Dune II* hubo que esperar hasta finales de 1994 para el lanzamiento del siguiente título del género *Warcraft: Orcs & Humans* de Blizzard Entertaiment. Aunque inferior a *Dune II* en aspectos como la diferenciación de los oponentes o la IA implementada, que fue considerada por muchos críticos como “trampas” [17], *Warcraft* fue el primer RTS en disponer de capacidades multijugador y un generador de mapas aleatorio, características claves en el futuro del género. No fue un éxito de ventas pero si vendió lo suficiente para que Blizzard siguiese desarrollando juegos RTS.

Westwood sin embargo sí que consiguió un gran éxito con su siguiente RTS, *Command & Conquer*, lanzado a finales de 1995. El juego cuenta la historia de la batalla entre la Iniciativa de Defensa Global (GDI) y la Hermandad de Nod. A diferencia de *Warcraft*, cada bando tiene unidades distintas con diferentes capacidades. Por ejemplo, mientras los GDI tienen como ventaja su potencia de ataque, la Hermandad de Nod tiene como característica principal la velocidad más que la fuerza. El juego incluye una gran variedad de unidades por lo que requiere un gran componente estratégico.

A pesar de su éxito *Command & Conquer* se encontró casi de inmediato con un gran rival, *Warcraft II* de Blizzard, lanzado en diciembre de 1995. Este juego mejoraba a su predecesor drásticamente

al mostrar mejores gráficos y una IA más sofisticada. Las ventas fueron muy superiores a las del título anterior y hasta llegaron a sacar 4 años más tarde una versión compatible con los servidores de partidas *on-line* de Blizzard, Blattle.net.

La serie de *Command & Conquer* por su parte continuó en 1996 con el lanzamiento de *Command & Conquer: Red Alert* (Figura 12), que presentaba una historia alternativa a la Segunda Guerra Mundial, donde un científico del futuro mata a Hitler antes de su ascenso al poder lo que acaba con un enfrentamiento entre aliados y soviéticos. El juego perfecciona el trabajo realizado anteriormente con bandos mucho más diferenciados y una interfaz más fácil de manejar. Consiguió una gran recepción entre la crítica y los jugadores lo que hizo que se lanzasen dos expansiones *Counterstrike* y *The Aftermath* y continuase como una saga independiente del original.

Figura 12. Imagen de Command & Conquer: Red Alert

El siguiente juego en evolucionar el género de los juegos de estrategia en tiempo real fue *Total Annihilation*, desarrollado por Cavedog Entertainment en 1997. Este juego fue uno de los primeros en presentar el diseño de las unidades en 3 dimensiones e innovó la interfaz en aspectos como la posibilidad de dar múltiples órdenes a nuestras unidades que serían ejecutadas en una secuencia, dando a los jugadores posibilidades de control hasta entonces únicamente disponibles en los juegos de estrategia por turnos.

Otro juego que evolucionó en género en su primera generación fue *Age of Empires* (Figura 13) de Ensemble Studios. Lanzado a finales de 1997, el juego presenta la idea de evolucionar tu civilización pasando de la Edad de Piedra hasta la Edad de Hierro siempre que se consigan alcanzar una serie de requisitos. En el juego se realizó un gran esfuerzo a la hora de desarrollar la IA, de forma que el ordenador se comportase de forma similar a un humano, evitando "hacer trampas", implementando estrategias complejas e incluso aprendiendo del comportamiento del jugador a lo largo de la partida [18].

Figura 13. Imagen de Age of Empires

El siguiente hito dentro del género llegó en 1998 de nuevo de la mano de Blizzard al lanzar el título en el que se centra este proyecto: *StarCraft*. Este juego, que será tratado con profundidad en el siguiente capítulo, se convirtió en el de más éxito de los inicios del género y supuso una gran evolución.

En los siguientes años la evolución del género se centró en el desarrollo de juegos con gráficos en 3 dimensiones siguiendo la estela de *Total Annihilation*. Los desarrolladores observaron que los ordenadores tenían ya la potencia para ejecutar juegos de estrategia en tiempo real en 3D y comenzaron el desarrollo de sus propios títulos. Juegos como *Emperator: Battle for Dune*, *Empire Earth* y *Shogun: Total War* fueron éxitos de esa época. En términos de IA destaca *Shogun: Total War*, ya que implementó en las tácticas del juego las estrategias del famoso libro "El arte de la guerra" de Sun Tzu [19].

Después de *Total Annihilation* o *StarCraft*, los fans se habían vuelto más exigentes y el género parecía estar estancado. Sin embargo, un nuevo por aquel entonces estudio de desarrollo llamado Relic Entertainment lanzó en septiembre de 1999 el videojuego *Homeworld* (Figura 14). Además de ser el primer juego con un entorno totalmente en 3 dimensiones, narraba una epopeya espacial bastante más elaborada que las habituales luchas entre el bien y el mal e implementaba formaciones de combate en 3 dimensiones. En lo que respecta a la IA, el juego incluye dificultad dinámica, lo que hace que, según progrese el jugador durante la partida, el enemigo se adapte a su habilidad. También destaca en la gestión de las formaciones de naves ya que no están limitadas en número y se adaptan según se van añadiendo componentes.

Figura 14. Imagen de Homeworld

WarCraft III: Reign of Chaos (Figura 15), lanzado en 2002 por Blizzard revolucionaria de estrategia en tiempo real. Sus desarrolladores definieron a este juego como un nuevo tipo de estrategia en tiempo real, RPS (*Role Playing Strategy* o estrategia de juego de rol en inglés). Las unidades en *WarCraft III* deben ser comandadas por una unidad especial o héroe (de manera similar al juego *Heroes of Might and Magic*). El juego añada dos nuevas razas con respecto a sus predecesores y cada raza posee recursos son distintos; por ejemplo, los muertos vivientes usan cadáveres como un recurso. *Warcraft III* también es conocido por el uso de trampas en los jugadores controlados por el ordenador.

Figura 15. Imagen de Warcraft III: Reign of Chaos

En los años más recientes los RTS se han enfocado principalmente en la acción con el lanzamiento de juegos como *Rome: Total War*, *Age of Empires III*, *Command and Conquer: Generals* o *El Señor de los Anillos: La Batalla por la Tierra Media* debido principalmente a razones comerciales. Con grandes expectativas también se han realizado juegos basados en las licencias de *Star Wars* (*Star Wars: Empire at War* de 2006), lanzados con resultados más bien modestos tanto en ventas como en lo que se refiere a avances del género.

El último RTS desarrollado a considerar es precisamente la segunda parte de *StarCraft*, *StarCraft II: Wings of Liberty* (Figura 16), lanzado en julio de 2010. Este juego se ha convertido en el RTS más vendido con más de 3 millones de unidades en el primer mes y mejora a su predecesor con una historia más elaborada, mejores gráficos y animaciones, nuevas unidades y acciones y mayor interacción con el entorno. Es este juego la IA esta implementada con un script más complejo que en anteriores juegos de Blizzard como *StarCraft* o *Warcraft III* ya que se han diseñado algoritmos para analizar lo que el jugador hace sin utilizar las típicas “trampas” de juegos anteriores con la excepción del modo de dificultad más alto, en el que la máquina puede ver todo lo que ocurre en el mapa [20].

Figura 16. Imagen de StarCraft II

2.3 StarCraft

2.3.1 Introducción

StarCraft (Figura 17) es un juego de estrategia en tiempo real de ciencia ficción militar lanzado para ordenadores personales el 31 de marzo de 1998. Ambientado en lo que sería el siglo 25 en la Tierra, el juego gira en torno a tres especies que luchan por el dominio del planeta Mar Sara, situado en una parte distante de la Vía Láctea conocida como el Sector Koprulu: los Terran, humanos descendientes de exiliados de la Tierra hábiles en adaptarse a cualquier situación; los Zerg, una raza de alienígenas insectoides en búsqueda de la perfección genética obsesionados con la asimilación de otras razas de la galaxia; y los Protoss, una sabia especie humanoide nativa del planeta Aiur que dispone de tecnología avanzada y habilidades psíquicas con las que tratan de preservar su civilización y su filosofía de vida, actualmente amenazada por los Zerg.

Figura 17. Imagen de StarCraft

La misión del jugador es construir una base y el ejército con el que atacar y vencer a sus oponentes (controlados por el ordenador o por humanos en modo multijugador). En el modo de un jugador algunas misiones tienen diferentes objetivos como defender una posición o evitar la muerte de alguno de los miembros de nuestro ejército mientras que en el modo multijugador el objetivo del juego es destruir todos los edificios enemigos. El jugador puede elegir una de las tres razas con las que jugar: Protoss, Zerg o Terran. Las tres razas tienen sus propias fortalezas y debilidades, así como unidades, habilidades y mecánicas de juego.

Por su parte, *StarCraft: Brood War* es el paquete de expansión de *StarCraft*. Lanzado el 30 de noviembre 1998, introdujo nuevas campañas, mapas, música, unidades adicionales para cada raza y mejoras. Las campañas retoman la historia desde el final del juego original, mientras que por su parte, la secuela *StarCraft II: Wings of Liberty*, lanzada en julio de 2010, continúa la historia a partir de la conclusión de *Brood War*.

En el año 2010 *StarCraft* fue reconocido como el juego de estrategia más vendido de la historia así como el videojuego con más permanencia en los deportes electrónicos [21].

2.3.2 Características del juego

Los aspectos a tener en cuenta a la hora de jugar a StarCraft son los siguientes:

Recursos

Como en la mayoría de juegos del género RTS, en StarCraft existen una serie de recursos disponibles que hay que gestionar a lo largo de la partida para construir o entrenar nuevas unidades.

- Minerales: Cristales azulados que se pueden encontrar a lo largo del mapa. Son el recurso principal para desarrollar nuevos edificios o unidades, de hecho los edificios más básicos solo requieren minerales para su construcción. En el caso de la raza Terran (la raza para la que se desarrolla el jugador automático del proyecto) es necesario el uso de trabajadores o VCEs para su recolección. Los minerales disponibles son limitados ya que cuando una veta se agota no se regenera y queda vacío el terreno donde se situaba por lo que es necesario la búsqueda de una nueva a lo largo del mapa.
- Gas vespeno: Gas de color verde que se puede encontrar en los géiseres repartidos por el mapa de juego. El vespeno se utiliza para desarrollar edificios y unidades de tecnología avanzada por lo que es importante en fases avanzadas del juego donde los jugadores ya han creado las unidades básicas y necesitan añadir habilidades nuevas para combatir a los enemigos. Para extraer vespeno en el caso de la raza Terran es necesaria la construcción de una refinería sobre un géiser con gas. Una vez construida la refinería el gas puede ser recolectado por VCEs. El gas vespeno no se agota de los géiseres pero, por el contrario, los géiseres tienen un límite de gas a partir del cual la velocidad de extracción se reduce de 8 unidades por VCE a 2 por lo que, cuando esto sucede, es recomendable la búsqueda de nuevos géiseres.
- Suministros: Representa la cantidad de comida disponible y por tanto el número de unidades que un jugador puede mantener. El desarrollo de casi cualquier unidad requiere el empleo de suministros los cuales están limitados a un máximo de 200 unidades en cualquier caso. La raza Terran puede incrementar el número de suministros disponibles construyendo Centros de mando (aumenta en 10 los suministros disponibles) o con Depósitos de Suministros, los cuales añaden 8 unidades a los suministros del jugador.

Razas

Existen 3 tipos de razas distintas en StarCraft, cada una de ellas con características distintas y diferentes edificaciones y unidades. La estrategia a emplear con cada raza debe ser distinta para adaptarse a las ventajas y debilidades de cada una de ellas.

- Terran: La raza que presenta más similitudes con los RTS clásicos. Son humanos de origen terrestre, descendientes de exiliados. Las unidades Terran pueden variar mucho de unas a otras y su relación coste-eficiencia es bajo en comparación a las otras razas. Destacan por las siguientes características:
    - Poder construir edificios en cualquier parte del mapa además de tener la posibilidad de trasladar la mayor parte de ello a otro lugar.
    - Tienen la habilidad de curar unidades con médicos o de reparar vehículos con trabajadores VCE.
    - Existe una gran variedad de unidades Terran que van desde los simples soldados a los potentes cruceros de batalla.
    - La mayoría de las unidades Terran pueden realizar ataques a distancia, destacando en esta característica los tanques de asedio.
    - A diferencia de otras razas, los edificios no se reparan lo que hace que si se encuentran ardiendo puedan echarse a perder si un VCE no los repara a tiempo.

- Zerg: Raza alienígena de individuos insectoides que actúan en grupo ya que el conjunto del ejército es guiado por el Superamo. Fue creada por los Xel'Naga, una raza ancestral de la que se desconoce su existencia en la actualidad, al igual que los Protoss. Su objetivo es la búsqueda de la perfección genética para lo que buscan especies más avanzadas por la galaxia para asimilar su información genética. A diferencia de los Terran no pueden edificar en cualquier parte del mapa ya que necesitan que la superficie se encuentre cubierta por biomateria para alimentar a las construcciones porque los edificios Zerg carecen de componentes mecánicos y necesitan de este elemento para "vivir". Otras características destacadas son:
    - El coste de producir unidades es el más bajo de las tres razas por lo que se puede disponer de un gran ejército en poco tiempo.
    - Se pueden regenerar automáticamente sin necesidad de médicos o mecánicos.

- Es la raza más rápida al desplazarse por el mapa.
- Por el contrario, las unidades Zerg son más débiles atacando solas aunque se compensa con la velocidad de producir nuevas unidades y el menor coste.
- Solo se necesita un edificio, el criadero, para producir unidades.
- La evolución de las larvas en unidades más complejas es lenta en comparación a unidades avanzadas de otras razas.
- Tienen la habilidad de esconderse bajo tierra lo que les hace indetectables. Esta habilidad puede utilizarse para realizar ataques sorpresa o para recuperarse.

- Protoss: Representa una raza muy avanzada, con grandes poderes físicos y psíquicos. Nativos del planeta Aiur, creían ser la raza más poderosa del universo hasta ser atacados por los Zerg, los cuales les superaban en número ampliamente. Disponen de la mejor tecnología de las tres razas del juego con ejemplos como el escudo protector del que disponen todas sus unidades y edificios. Otras de sus características principales son:
  - Los edificios no se construyen, se invocan, por lo que la sonda Protoss sólo tiene que invocar al edificio que se requiera y este se transportará desde el planeta Aiur sin más intervención de la sonda que, de esta forma, queda liberada y puede dedicarse a otras tareas.
  - Las zonas en las que se puede edificar están limitadas por el área de influencia del Pilón más cercano, ya que son estos los que transmiten la energía.
  - La dependencia del pilón hace de este un punto vulnerable ya que si se destruye los edificios que dependen de él quedan inutilizados.
  - El consumo de recursos es mayor que en las otras razas, no solo al crearse si no también al atacar.
  - Única raza que posee unidades permanentemente invisibles.
  - La creación de nuevas unidades es lenta en relación a otras razas.

2.3.3 Mecánica del juego

Existen dos tipos de modos de juego el de un solo jugador y el modo multijugador. En el modo de un jugador se puede elegir entre el modo campaña o jugar una partida personalizada. La campaña consiste en una serie de misiones divididas en 3 capítulos (más otros 3 incluidos en la expansión *Broodwar*), controlando en cada uno una raza distinta, en los que se han de cumplir diversas órdenes como proteger a unidades especiales, destruir un edificio enemigo, llevar ciertas unidades a algún lugar específico o simplemente aguantar el ataque enemigo. En las partidas personalizadas el jugador puede elegir el mapa, la raza con la que jugar y el número de enemigos a los que enfrentarse hasta un máximo de 7. En este modo se comienza la partida con un edificio principal y 4 jugadores. Es un modo muy similar al multijugador con la diferencia de que los enemigos están controlados por la computadora y es el modo en el que se emplea el jugador automático de este proyecto.

El modo multijugador permite jugar contra un máximo de 7 oponentes humanos a través de una red LAN o con Internet utilizando los servidores de Battle.net. En este modo existen distintos tipos de partidas entre las que destacan las siguientes:

- Cuerpo a cuerpo: El jugador debe de eliminar al resto de oponentes.
- Escalera: Se determina una clasificación de los jugadores.
- Capturar la bandera: El objetivo es capturar todas las banderas del jugador contrario para ganar.
- Muerte súbita: Gana el primer jugador que destruya el edificio principal del enemigo.
- Matanza: El jugador debe de destruir el mayor número de unidades y edificios enemigos para ganar.
- Avaricia: El objetivo en esta partida es la de llegar a acumular una cantidad determinada de minerales o gas vespeno.

En *StarCraft*, como en la mayor parte de RTS, los jugadores deben desarrollar una estrategia que permita gestionar de forma rápida y eficaz su economía, la tecnología y el ejército del que disponen con el fin de derrotar a su oponente. La base de juego es organizar la recolección de recursos y el uso de dichos recursos para la construcción de edificios, investigación de mejoras y entrenar nuevas unidades. Sin embargo, existen muchas interrelaciones y características que hacen que *StarCraft* sea mucho más complejo.

Entre las propiedades del juego más importantes a tener en cuenta están las siguientes:

- Cada edificio o unidad requiere un número determinado de recursos y disponer del edificio o tecnología necesarios para su desarrollo.
- En el juego hay "niebla de guerra" lo que hace que no se pueda disponer de información del mapa a partir de cierta distancia de las unidades del jugador.
- La estrategia debe de variar según la raza elegida que disponen de unidades y edificios con características muy distintas.
- Incrementar el número de unidades es importante hasta cierto punto ya que alcanzado un nivel de desarrollo es más importante investigar nuevas tecnologías que hagan a nuestras unidades más potentes e incorporar nuevas habilidades.
- Durante las batallas hay que saber elegir si es mejor dejar de invertir recursos tratando de vencer al enemigo para centrarse en las siguientes batallas o por el contrario si hay posibilidades de vencer.

2.4 Inteligencia Artificial

Si se busca IA en un diccionario probablemente se encontrará algo parecido a lo siguiente: “La capacidad de un ordenador u otra máquina para llevar a cabo tareas que normalmente se considera que requieren inteligencia” [23]. John McCarthy, fue en primero en definir la IA en una conferencia de la Universidad de Darmouth en 1965 donde planteó que "Este estudio procederá sobre la base de la conjetura de que todos los aspectos del aprendizaje o cualquier otro rasgo de la inteligencia pueden, en principio, ser descritos de una forma tan precisa que se puede crear una máquina que los simule" [24].

Sin embargo otras fuentes definen la inteligencia artificial como la rama de la ciencia de la computación que estudia la resolución de problemas no algorítmicos mediante el uso de cualquier técnica de computación disponible, sin tener en cuenta la forma de razonamiento subyacente a los métodos que se apliquen para lograr esa resolución [25].

Desde otra perspectiva, se puede pensar en la IA como el intento por desarrollar una tecnología capaz de proveer al ordenador capacidades de razonamiento similares a los de la inteligencia humana o, desde otro enfoque distinto, como la investigación relativa a los mecanismos de la inteligencia humana que se emplean en la simulación de validación de teorías.

La conclusión es que la definición de IA es bastante amplia y flexible.

2.4.1 Inicio de la Inteligencia Artificial

Durante la Segunda Guerra Mundial, el prestigioso matemático británico Alan Turing y el experto en computación estadounidense Claude Shannon trabajaron juntos descifrando los códigos secretos usados por el ejército nazi con la máquina Enigma y con los codificadores de teletipos FISH [22]. Fruto de esta unión, Turing y Shannon junto a otros matemáticos como como Alonzo Church y Kurt Gödel establecieron las bases de la teoría de la computación, donde señalaban la IA como el campo más importante hacia el que había que dirigir todos los esfuerzos de investigación en un futuro.

Gracias a estas investigaciones en el campo de la IA, en marzo del 1950 Claude Shannon presentó junto al economista y matemático D. G. Champernowne un artículo llamado “*Programming a Computer for Playing Chess*” en la revista *Philosophical Magazine*, en dicho artículo se

especificaban las primeras técnicas y algoritmos necesarios para crear un programas de ajedrez, pero en ese momento no había ningún ordenador con el suficiente potencial para poder ejecutarlo, así que no fue hasta el noviembre de 1951 que el Dr. Dietrich Prinz pudo escribir el programa original en una Ferranti Mark I, la primera computadora electrónica comercial.

En el año 1952 pudieron por fin poner a prueba su programa, aunque fue simulando los movimientos de la computadora porque el programa no mostraba gráficos. En su estreno, la máquina perdió su primera partida frente a Alick Glennie (un amigo de Alan Turing), pero ganó la segunda partida frente a la esposa de Chapernowe. Con este programa, Shannon y D. G. Champernowne sentaron las bases prácticas para la creación de los programas de ajedrez que se siguen aplicando hoy en día, aunque con la diferencia que en esos tiempos el ordenador podía tardar entre 15 y 20 minutos para procesar cada movimiento. Al ser una partida simulada y no disponer de gráficos, tampoco se puede considerar como el primer videojuego, aunque tiene un enorme valor cara a la evolución de este medio y puede considerarse el primer antecedente de IA aplicada a los videojuegos.

2.4.2 Inteligencia Artificial en los videojuegos

En el desarrollo de videojuegos el objetivo de la IA se centra en conseguir que los personajes tengan comportamientos similares a los humanos. La IA se utiliza en una amplia variedad de campos muy dispares dentro de un juego. La más evidente es en el control de los personajes no jugables a pesar de que actualmente el medio más común de control es la secuencia de comandos.

La búsqueda de caminos o *Pathfinding* es otro de uso común de la IA ampliamente visto en juegos como los del género RTS. El *Pathfinding* es el método para determinar cómo llevar un personaje no controlable de un punto en un mapa a otro, teniendo en cuenta el terreno, obstáculos y, posiblemente, "niebla de guerra", evitando las colisiones con otras entidades (otros personajes, jugadores...) o interactuando con ellos. Una buena implementación de esta técnica guiará a los personajes por el camino más corto o rápido posible.

Otro uso de la IA dentro de los videojuegos es la toma de decisiones: la capacidad de decidir qué acción realizar. Para implementar la toma de decisiones la mayoría de los desarrolladores utilizan máquinas de estado o árboles de decisión aunque algunos juegos siguen empleando IA basada en reglas [26]. La toma de decisiones en el género de los RTS tiene una complejidad añadida ya que,

por norma general, hay que tener en cuenta la situación global del ejército del jugador y no solo de una unidad.

El concepto de aprendizaje ha sido recientemente explorado en juegos como *Black & White* donde las criaturas del juego son capaces de "aprender" de las acciones tomadas por el jugador y su comportamiento se modifica en consecuencia.

Para la comunidad de desarrolladores de videojuegos, el premio para el juego con mejor IA del año 2014 fue para *Middle Earth: Shadow of Mordor* de Monolith Studios por el trabajo hecho con planificadores [27].

2.4.3 Técnicas de IA empleadas en videojuegos

2.4.3.1 Movimiento

Una de las funciones básicas de la IA en los videojuegos es mover los personajes no controlados por el jugador. Hasta los personajes controlados por IA más antiguos (fantasmas del PacMan o el palo contrario del Pong) tenían algoritmos de movimiento que no son tan distintos de los juegos que se desarrollan en la actualidad [Ian Millington, 2006].

Todos los algoritmos de movimiento tienen la misma forma básica: toman los datos acerca de su estado y el estado del mundo, y obtienen una salida representando el movimiento que deben realizar.

Algunos algoritmos de movimiento requieren muy pocos parámetros de entrada como, por ejemplo la posición del personaje y del enemigo al que perseguir. Otros requieren mucha interacción con el estado del juego y la geometría del nivel. Un algoritmo de movimiento que evita chocarse con paredes, por ejemplo, necesita tener acceso a la geometría de la pared para comprobar posibles colisiones potenciales.

La salida también puede variar. En la mayor parte de juegos es normal tener algoritmos de movimiento que den como resultado la velocidad deseada. Por ejemplo, si un personaje ve a su enemigo a su izquierda, su reacción debería ser dirigirse a la izquierda a toda velocidad. A menudo, en juegos antiguos, sólo había dos velocidades: máxima y estacionaria (sin moverse), donde en algunos casos toca añadir la velocidad normal de caminar. Por tanto, la única salida o

resultado a obtener era la dirección en la que moverse. Esto se llama movimiento cinemático; no tiene en cuenta cómo los personajes aceleran o aminoran su velocidad.

Recientemente, se ha tomado mucho interés en los comportamientos de giro: no son cinemáticos, pero sí dinámicos. Estos algoritmos tienen en cuenta la velocidad instantánea del personaje, donde típicamente el algoritmo debe saber la velocidad actual del personaje, así como su posición. Los resultados que ofrece un algoritmo dinámico son fuerzas o aceleraciones, con el objetivo de cambiar la velocidad del personaje.

2.4.3.2 Búsqueda de caminos

La búsqueda de caminos puede ser usada simplemente para ver en qué dirección se ha de mover el personaje, mientras que otra parte de la IA decida cuál es ese objetivo al que debe aproximarse. También hay otro tipo de IA que fusiona ambos papeles: *goal pathfinding*, el cual se dedica tanto a decidir un objetivo como a indicar cómo dirigirse hacia él.

La solución obtenida tiene que ser la más sencilla y directa. No tiene sentido que un personaje que está en la habitación anexa a la que tiene la puerta que da a parar hacia donde se quiere dirigir, de una vuelta por todas las habitaciones de la casa hasta dar con la puerta deseada.

En cuanto a implementación, la mayoría de juegos usan soluciones de búsqueda de caminos basadas en el algoritmo A*, un algoritmo de búsqueda de grafos que encuentra el camino de coste mínimo a partir de un nodo inicial y un nodo destino. Se trata de un algoritmo heurístico, ya que una de sus principales características es que hace uso de una función de evaluación heurística, mediante la cual etiqueta los diferentes nodos de la red y que determina la probabilidad de dichos nodos de pertenecer al camino óptimo [28]. Para etiquetar los distintos nodos se emplea la siguiente función:

$$\mathbf{f(n)= g(n) + h(n)}$$

donde **g(n)** indica la distancia real del camino recorrido para llegar a n, desde el nodo inicial y **h(n)** representa el valor heurístico desde n hasta el destino.

2.4.3.3 Toma de decisiones

Las técnicas para la toma de decisiones persiguen que el personaje procese el conjunto de la información que dispone para que genere una acción en consecuencia. La información disponible se puede dividir en dos grupos: conocimiento externo, y conocimiento interno. El primero es el procedente del entorno del juego que rodea al jugador (posición de enemigos, posición donde se encuentra, dirección del ruido...), y el segundo es información interna del personaje (salud, objetivos). A partir de esta información la técnica de toma de decisiones empleada debe ayudar al personaje a discernir entre la mejor opción a realizar de las disponibles en función de su objetivo.

2.4.3.4 Árboles de decisión

Un árbol de decisión consiste en un conjunto de condiciones organizadas jerárquicamente mediante nodos y aristas de forma que, partiendo del nodo raíz, se puede llegar a una decisión final mediante las condiciones que contiene cada nodo.

Los árboles de decisión son rápidos, fácilmente implementables y sencillos de entender. Aunque pertenezcan al tipo de técnicas de toma de decisión más sencillas, pueden resultar bastante sofisticados con algunas extensiones/variaciones.

Un árbol de decisión clasifica instancias caracterizadas como un conjunto de atributos [29]. Cada nodo no terminal del árbol representa un atributo y tiene tantas aristas como valores puede tomar dicho atributo. Los nodos terminales representan clases por lo que para clasificar una instancia se comienza por el nodo raíz y se continúa por las aristas que marquen el valor que tomen los atributos hasta llegar a un nodo terminal, que indica la clasificación de la instancia. En la imagen a continuación se representa un árbol con el que decidir si se concede o no un préstamo:

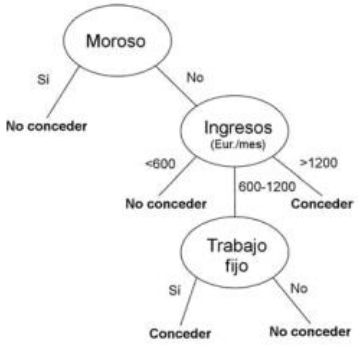

Figura 18. Ejemplo de árbol de decisión

2.4.3.5 Máquinas de estado

Es una de las técnicas más utilizadas en la IA de videojuegos. Una máquina de estados es una estructura de programa que sirve para determinar el comportamiento de un personaje en función del estado en el que se encuentre. Un estado debe representar una situación concreta del personaje (o ejército en un RTS) dentro del juego como por ejemplo ser atacado. Para cada estado por tanto se debe definir un conjunto de acciones a tomar así como las condiciones para transitar a otro estado. Los distintos estados están conectados mediante transiciones, que se producen solamente si se cumplen unas determinadas condiciones asociadas para el paso de un estado a otro.

En la figura 19 se puede observar un ejemplo de máquina de estado:

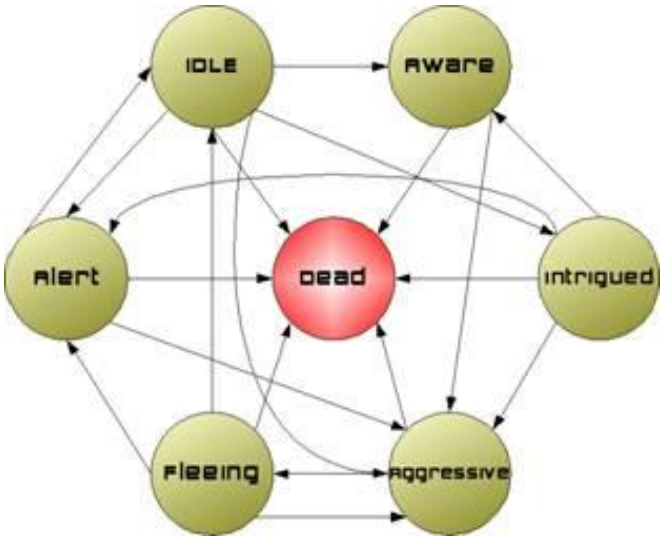

Figura 19. Ejemplo de máquina de estados [30]

Capítulo 3: Descripción del sistema

3.1 Introducción

El sistema desarrollado para este proyecto implementa un jugador automático para el videojuego *StarCraft: Broodwar* a través de la API (siglas de *Application Programming Interface*, interfaz de programación de aplicaciones en inglés) BWAPI. Para ello se ha desarrollado una máquina de estados combinada con árboles de decisión con el objetivo de que el jugador automático sea capaz de ganar a su oponente en el juego. El estado del jugador se representa mediante la variable **Estado**, la cual varía indica la siguiente transición a realizar. Realizada la transición al estado especificado se tienen en cuenta una serie de condiciones que determinan la acción a realizar o la siguiente transición a un nuevo estado.

3.2 Análisis del sistema

3.2.1 Descripción de las características funcionales

El sistema desarrollado para este proyecto implementa un jugador automático para el videojuego *StarCraft: Broodwar* a través de la API (siglas de *Application Programming Interface*, interfaz de programación de aplicaciones en inglés) BWAPI. Para ello se ha desarrollado una máquina de estados combinada con árboles de decisión con el objetivo de que el jugador automático sea capaz de ganar a su oponente en el juego. El estado del jugador se representa mediante la variable **Estado**, la cual varía indica la siguiente transición a realizar. Realizada la transición al estado especificado se tienen en cuenta una serie de condiciones que determinan la acción a realizar o la siguiente transición a un nuevo estado.

Para conseguir su misión, el jugador debe gestionar las unidades para que realicen tareas de recolección de minerales y gas, creación de nuevas estructuras, entrenamiento de soldados, fabricación de vehículos y ataques y defensa de los oponentes. Concretamente, el jugador automático desarrollado realiza las funciones de:

- Recolección de mineral
- Recogida de gas vespeno
- Construcción de refinerías de gas vespeno

- Construcción de depósitos de suministros
- Construcción de barracas
- Construcción de fábricas
- Construcción de taller de maquinaria
- Construcción de puerto estelar
- Construcción de academia
- Entrenamiento de VCE
- Entrenamiento de soldados
- Entrenamiento de médico
- Entrenamiento de murciélago de fuego
- Fabricación de buitre
- Fabricación de tanque de asedio
- Fabricación de espectro
- Investigación de stim pack
- Investigación de tanque de asedio
- Investigación de minas
- Defender edificios o vehículos de ataques enemigos
- Atacar enemigos

3.2.2 Restricciones del sistema

El hardware mínimo necesario para la realización de este proyecto es:

- Ordenador con las siguientes características:
  - Procesador Pentium III a 800Mhz o equivalente
  - 512 MB de memoria RAM
  - una tarjeta gráfica con 64 megabytes.

El software necesario para el proyecto es:

- Sistema operativo Windows XP SP3
- Java Development Kit 8.0
- Eclipse
- StarCraft: Broodwar v1.16.1

- JNI-BWAPI: Un framework de programación que permite interactuar mediante programas escritos en Java con BWAPI (iniciales de *The Brood War Application Programming Interface*), un entorno que a su vez interactúa con *StarCraft: Broodwar* creado en C++.
- ChaosLauncher: inyector del código desarrollado en el juego

3.2.3 Entorno operacional

El entorno en el que se ha desarrollado el proyecto es el siguiente:

- Ordenador con las siguientes características:
  - Procesador Intel Core i3 2,27 Ghz
  - 4 GB de memoria RAM
  - Tarjeta gráfica ATI Mobility Radeon HD 5470 con 512MB dedicados
- Sistema operativo Windows 7/10
- Java Development Kit 5.0
- Eclipse Kepler
- StarCraft: Broodwar v1.16.1
- BWAPI 3.7.4
- JNI-BWAPI 1.0
- ChaosLauncher v0.5.4.1

3.2.4 Especificación de casos de uso

La figura 20 presenta el diagrama de casos de uso de este sistema.

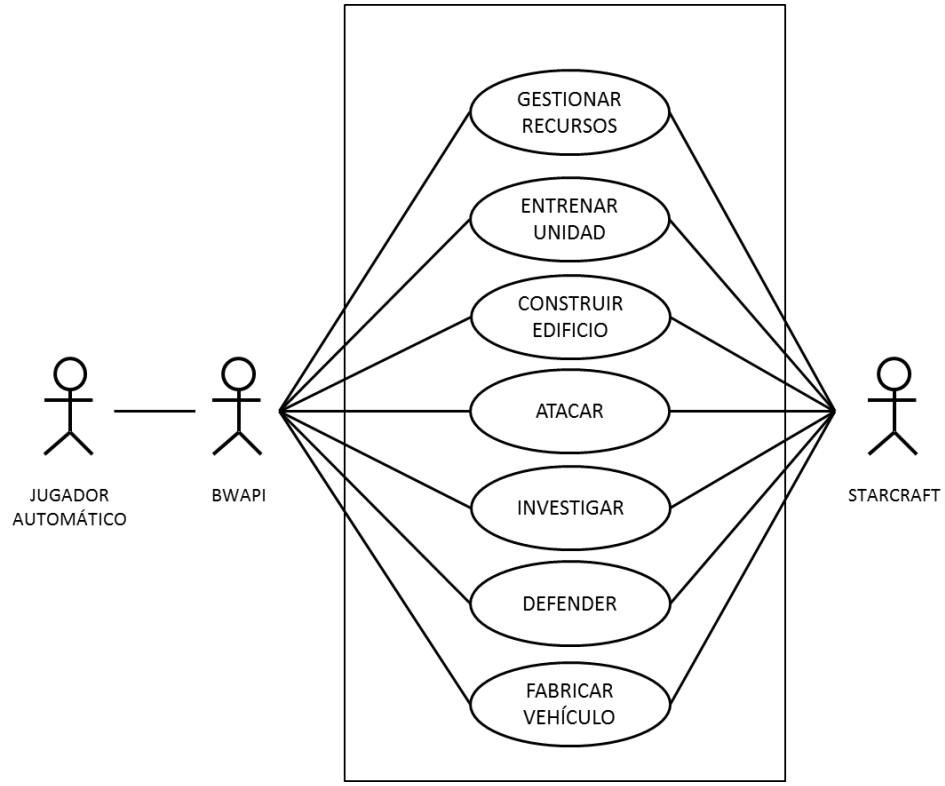

Figura 20. Diagrama de casos de uso del sistema

3.2.4.1 Descripción de los actores

- *StarCraft: Broodwar*: Videojuego sobre el que funciona el autómata que se desarrolla en este proyecto.
- BWAPI: Entorno de código abierto para interactuar con *Starcraft: Broodwar*
- Jugador Automático: Jugador implementado mediante una máquina de estados, capaz de jugar al juego por sí mismo.

3.2.4.2 Descripción de los atributos de los casos de uso

A continuación se realiza una descripción del significado de cada uno de los atributos utilizados para la descripción de los casos de uso:

- Código: Identificación unívoca abreviada del caso de uso, se construye mediante CU seguido de un - y de tres dígitos. Por ejemplo CU-001.

- Nombre: Identificación extendida del caso de uso.
- Actores: Conjunto de entidades que interactúan con el caso de uso. El caso de uso representa una funcionalidad demandada por un actor.
- Descripción: Se realiza una descripción básica de la funcionalidad o funcionalidades del caso de uso.
- Precondiciones y poscondiciones: Se realiza una descripción de las condiciones que deben cumplirse para poder realizar una operación, y el estado en el que queda el sistema tras realizar una operación.
- Escenario: Se realiza una descripción básica de las acciones que se ejecutan paso a paso en el caso de uso.

3.2.4.3 Descripción textual de los casos de uso

Gestionar Recursos	
Código	CU-001
Nombre	Gestionar Recursos
Actores	Starcraft, BWAPI
Descripción	Comprueba que existen recursos suficientes para construir edificios, entrenar unidades o fabricar vehículos
Precondiciones	Inicio del juego o falta de recursos para acometer una acción
Poscondiciones	Se ordena la recolección de minerales y/o gas a los VCEs disponibles o se entrenan más si es necesario.
Escenario	Comprobación de número de VCEs del que dispone el jugador. Si no es suficiente para el tamaño del ejército se ordena entrenar más. Si el número de VCEs es suficiente, se le asignará a cada uno recolectar minerales o gas en función de lo almacenado. Si no se dispone de refinería, se ordenará su construcción si se cumplen los requisitos para ello.

Tabla 1 - Caso de uso CU-001

Entrenar Unidad	
Código	CU-002
Nombre	Entrenar Unidad
Actores	Starcraft, BWAPI
Descripción	Comprueba que existen recursos suficientes para construir edificios, entrenar unidades o fabricar vehículos
Precondiciones	Existen los edificios necesarios y los recursos suficientes
Poscondiciones	Se ordena entrenar a la unidad correspondiente.
Escenario	Se estima necesario entrenar nueva unidad por lo que se comprueba si están construidos los edificios necesarios para entrenar a la unidad y se dispone de los recursos suficientes. Si es así, se ordena iniciar el entrenamiento.

Tabla 2 - Caso de uso CU-002

Construir Edificio	
Código	CU-003
Nombre	Construir Edificio
Actores	Starcraft, BWAPI
Descripción	Busca un lugar adecuado para construir el edificio cerca de la base del jugador y si se encuentra y se tienen los recursos se construye el edificio.
Precondiciones	El jugador tiene algún VCE disponible, un lugar adecuado para construir, recursos suficientes y cumple los requisitos previos.
Poscondiciones	El VCE inicia la construcción del edificio.
Escenario	Se necesita construir un nuevo edificio para lo cual se busca un VCE libre, se busca un emplazamiento adecuado y se comprueba si hay recursos suficientes y si se cumplen las condiciones previas necesarias para construir ese edificio. Si se cumplen todos los requisitos se ordena al VCE identificado que comience la construcción del edificio.

Tabla 3 - Caso de uso CU-003

Atacar	
Código	CU-004
Nombre	Atacar
Actores	Starcraft, BWAPI
Descripción	Selecciona un batallón de unidades y les ordena atacar unidades enemigas
Precondiciones	El jugador dispone de unidades suficientes para atacar.
Poscondiciones	Las unidades atacan al enemigo.
Escenario	El jugador tiene disponibles unidades suficientes para completar batallones y no está recibiendo ataques enemigos. Se selecciona una unidad enemiga para ser atacada y se ordena atacar a todas las unidades de un batallón a la unidad enemiga seleccionada.

Tabla 4 - Caso de uso CU-004

Investigar	
Código	CU-005
Nombre	Investigar
Actores	Starcraft, BWAPI
Descripción	Selecciona edificio para investigar mejoras en los vehículos que fabrica
Precondiciones	El jugador dispone del edificio y de los recursos necesarios para iniciar la investigación.
Poscondiciones	Las unidades incorporan la tecnología investigada.
Escenario	El jugador dispone del edificio indicado para realizar la investigación, se comprueba que no se dispone de la mejora requerida y si de los recursos necesarios. Si se cumplen todos los requisitos se ordena al edificio correspondiente iniciar la investigación de la mejora.

Tabla 5 - Caso de uso CU-005

Defender	
Código	CU-006
Nombre	Defender
Actores	Starcraft
Descripción	Selecciona un batallón de unidades y las desplaza hasta la unidad atacada
Precondiciones	Una unidad del jugador es atacada.
Poscondiciones	Las unidades de un batallón acuden a atacar al enemigo.
Escenario	Una unidad del jugador está recibiendo un ataque del enemigo. Se seleccionan todos los batallones completados y se ordena desplazar a todas las unidades hasta la unidad atacada.

Tabla 6 - Caso de uso CU-006

Fabricar	
Código	CU-007
Nombre	Fabricar
Actores	Starcraft
Descripción	Comprueba los requisitos necesarios para construir el vehículo requerido.
Precondiciones	El jugador dispone recursos y edificios adecuados para fabricar el vehículo.
Poscondiciones	Se ordena la fabricación del vehículo.
Escenario	El jugador tiene disponibles unidades suficientes para completar batallones y no está recibiendo ataques enemigos. Se selecciona una unidad enemiga para ser atacada y se ordena atacar a todas las unidades de un batallón a la unidad enemiga seleccionada.

Tabla 7 - Caso de uso CU-007

3.2.4.2 Descripción de los atributos de los requisitos

Para la realización de la descripción textual de los distintos requisitos que han sido identificados, se han seleccionado una serie de atributos que describen cada uno de los requisitos:

- Código: Identificación unívoca abreviada del requisito, se construye mediante el código del requisito seguido de un - y de tres dígitos. Los requisitos serán divididos en funcionales y no funcionales y sus códigos son RF para los requisitos funcionales y RNF para los requisitos no funcionales. Por ejemplo RF-001.
- Nombre: Identificación extendida del requisito.
- Descripción: Se realiza una descripción básica del requisito que ha sido identificado.
- Fuente: Indica a través de qué fuente ha sido identificado el requisito. Normalmente este valor se corresponderá con uno o varios códigos de los casos de uso.
- Necesidad: Determina el grado de implementación del requisito. Los valores que puede tomar este atributo son los siguientes:
    - Esencial: El requisito tiene que ser implementado.
    - Deseable: Es preferible implementar el requisito, pero no es obligatorio.
    - Opcional: El requisito se podrá implementar, pero no es importante ni obligatorio.
- Prioridad: Define la importancia del requisito, de forma que permita definir el orden en el cuál será incluido en el proceso de diseño y el orden de implementación. Los valores que puede tomar este atributo son los siguientes:
    - Alta: El requisito debe ser implementado en las fases iniciales del desarrollo.
    - Media: El requisito debe ser implementado una vez que hayan sido implementados los requisitos de prioridad alta.
    - Baja: El requisito debe ser implementados en las fases finales del desarrollo. Estos requisitos no influirán en el correcto funcionamiento del sistema.
- Estabilidad: Define la estabilidad de los requisitos durante la vida útil del software. Esto implica si los requisitos podrá ser o no modificado durante el ciclo del vida. Los valores que puede tomar este atributo son los siguientes:
    - Estable: El requisito no puede variar durante el ciclo de vida del sistema.
    - Inestable: El requisito puede variar a lo largo de la ciclo de vida del sistema.

- Verificabilidad: Define el grado de verificabilidad de un requisito, es decir indica en qué grado es posible comprobar que el requisito se ha incorporado en el sistema desarrollado. Los valores que puede tomar este atributo son los siguientes:
    - Alta: Se puede verificar que el requisito ha sido implementado en el sistema. Este tipo de requisitos se corresponden con las funcionalidades básicas del sistema.
    - Media: Se puede verificar que el requisito ha sido implementado en el sistema. Pero requiere de una comprobación compleja o del código fuente del sistema.
    - Baja: Es difícil verificar si el requisito ha sido implementado en el sistema o en algunos casos no es posible.

3.2.5 Especificación de requisitos

Configuración de Chaoslauncher			
Código	**RF-001**	**Fuente**	CU-009
Nombre	Configuración de Chaoslauncher		
Descripción	Configuración correcta de Chaoslauncher		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 8 – Requisito del Sistema RF-001

BWAPI			
Código	**RF-002**	**Fuente**	CU-008 CU-009
Nombre	BWAPI		
Descripción	La versión de BWAPI utilizada será la 3.7		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 9 – Requisito del Sistema RF-002

Entrenar VCE			
Código	**RF-003**	**Fuente**	CU-001 CU-002
Nombre	Entrenar VCE		
Descripción	El autómata podrá entrenar los VCEs requeridos		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 10 – Requisito del Sistema RF-003

Construir Refinería			
Código	**RF-004**	**Fuente**	CU-001 CU-003
Nombre	Construir Refinería		
Descripción	El autómata podrá construir refinerías de gas vespeno		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 11 – Requisito del Sistema RF-004

Construir Barracas			
Código	**RF-005**	**Fuente**	CU-001 CU-003
Nombre	Construir		
Descripción	El autómata podrá construir barracas		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 12 – Requisito del Sistema RF-005

Construir Depósitos			
Código	**RF-006**	**Fuente**	CU-001 CU-003
Nombre	Construir Depósitos		
Descripción	El autómata podrá construir depósitos de suministros		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 13 – Requisito del Sistema RF-006

Construir Fábrica			
Código	**RF-007**	**Fuente**	CU-001 CU-003
Nombre	Construir Fábrica		
Descripción	El autómata podrá construir fábricas Terran		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 14 – Requisito del Sistema RF-007

Entrenar Soldados			
Código	**RF-008**	**Fuente**	CU-001 CU-002
Nombre	Entrenar soldados		
Descripción	El autómata podrá entrenar soldados		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 15 – Requisito del Sistema RF-008

Entrenar Médicos			
Código	**RF-009**	**Fuente**	CU-002
Nombre	Entrenar médicos		
Descripción	El autómata podrá entrenar médicos		
Necesidad	Deseable	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 16 – Requisito del Sistema RF-009

Entrenar Murciélagos de fuego			
Código	**RF-010**	**Fuente**	CU-001 CU-002
Nombre	Entrenar murciélagos de fuego		
Descripción	El autómata podrá entrenar murciélagos de fuego		
Necesidad	Deseable	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 17 – Requisito del Sistema RF-010

Fabricar Tanques			
Código	**RF-011**	**Fuente**	CU-007
Nombre	Fabricar tanques		
Descripción	El autómata podrá fabricar tanques		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 18 – Requisito del Sistema RF-011

Detectar Ataques			
Código	**RF-012**	**Fuente**	CU-006
Nombre	Detectar ataques		
Descripción	El autómata podrá detectar ataques enemigos.		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 19 – Requisito del Sistema RF-012

Atacar			
Código	**RF-013**	**Fuente**	CU-006 CU-004
Nombre	Atacar		
Descripción	El autómata podrá detectar enemigos para atacarlos.		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 20 – Requisito del Sistema RF-013

Investigar			
Código	**RF-014**	**Fuente**	CU-005
Nombre	Atacar		
Descripción	El autómata podrá investigar nuevas tecnologías para aplicar en vehículos o unidades.		
Necesidad	Deseable	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 21 – Requisito del Sistema RF-014

Gestionar Batallón			
Código	**RF-015**	**Fuente**	CU-003 CU-004 CU-006 CU-007
Nombre	Gestionar batallón		
Descripción	El autómata agrupará sus unidades y vehículos con el fin de realizar ataques más eficaces		
Necesidad	Deseable	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 22 – Requisito del Sistema RF-015

Gestionar Bajas			
Código	**RF-016**	**Fuente**	CU-001 CU-003 CU-004
Nombre	Gestionar bajas		
Descripción	El autómata gestionará los vehículos y unidades que han sido eliminados para poder llevar un inventario de las unidades disponibles y gestionar correctamente los batallones.		
Necesidad	Deseable	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 23 – Requisito del Sistema RF-016

Fabricar Espectros			
Código	**RF-017**	**Fuente**	CU-007
Nombre	Fabricar espectros		
Descripción	El autómata será capaz de fabricar naves espectro.		
Necesidad	Deseable	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 24 – Requisito del Sistema RF-017

Construir Academia			
Código	**RF-018**	**Fuente**	CU-002
Nombre	Construir Academia		
Descripción	El autómata podrá construir una academia Terran.		
Necesidad	Deseable	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 25 – Requisito del Sistema RF-018

Construir Puerto Estelar			
Código	**RF-019**	**Fuente**	CU-001 CU-003
Nombre	Construir Puerto Estelar		
Descripción	El autómata podrá construir puerto estelar.		
Necesidad	Deseable	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 26 – Requisito del Sistema RF-019

3.3 Diseño del sistema

3.3.1 Arquitectura del sistema

La arquitectura del sistema se basa en una máquina de estados implementados mediante árboles de decisión donde se ejecutan las acciones a realizar por el jugador automático. En la figura 21 se detalla el diseño de la máquina de estados diseñada en este proyecto, donde cada uno de los estados de la máquina se desarrolla mediante un árbol de decisión.

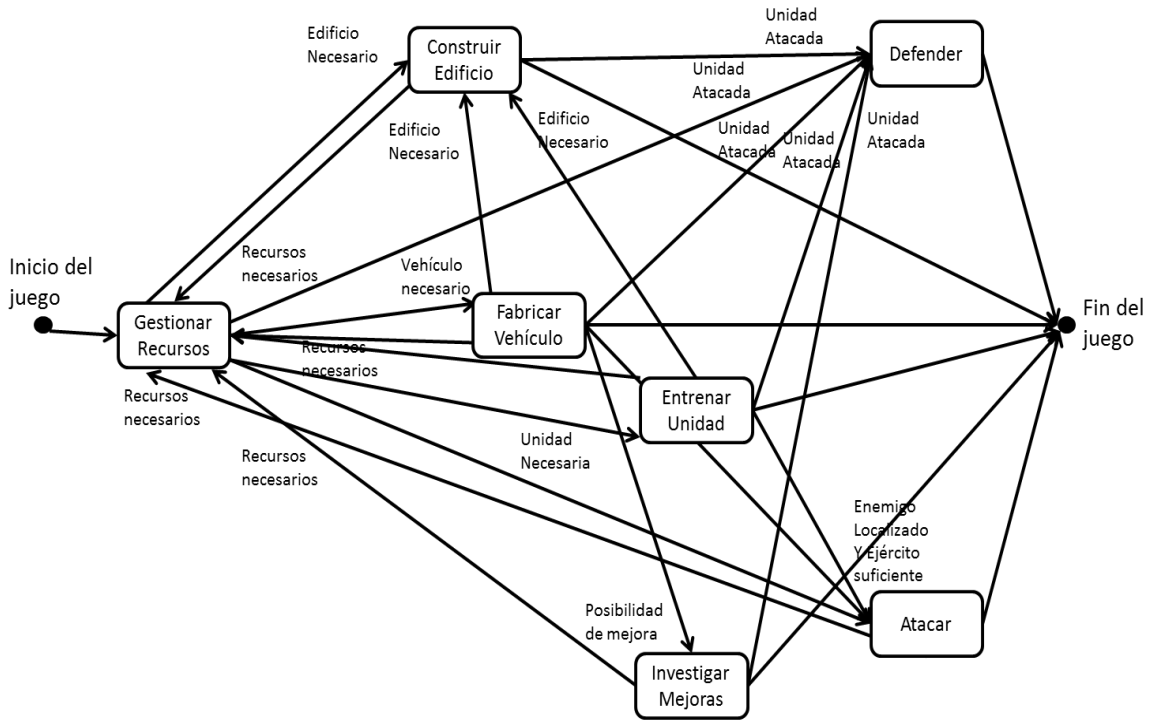

Figura 21. Diagrama de la máquina de estados del sistema

En la figura 22 se muestra el árbol de decisión del estado Gestionar recursos en el que se incluyen todas las acciones relacionadas con la recogida de minerales y gas vespeno. En caso de requerir la construcción de un edificio o el entrenamiento de nuevas unidades (como entrenar un VCE) se transita al estado correspondiente.

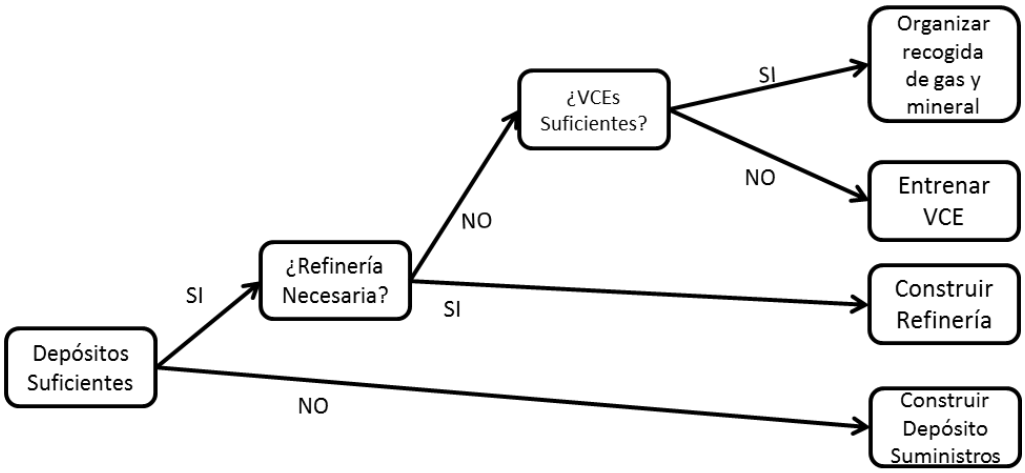

Figura 22. Diagrama del árbol del estado "Gestionar Recursos"

La figura 23 presenta el diseño del árbol incluido en el estado entrenar unidad. Tanto los nodos marcados con asterisco así como el nodo atacar transitan a otro estado. Una vez creada la unidad se integrará en un batallón, el cual deberá disponer además de al menos un médico siempre que se tenga la capacidad de entrenarlos.

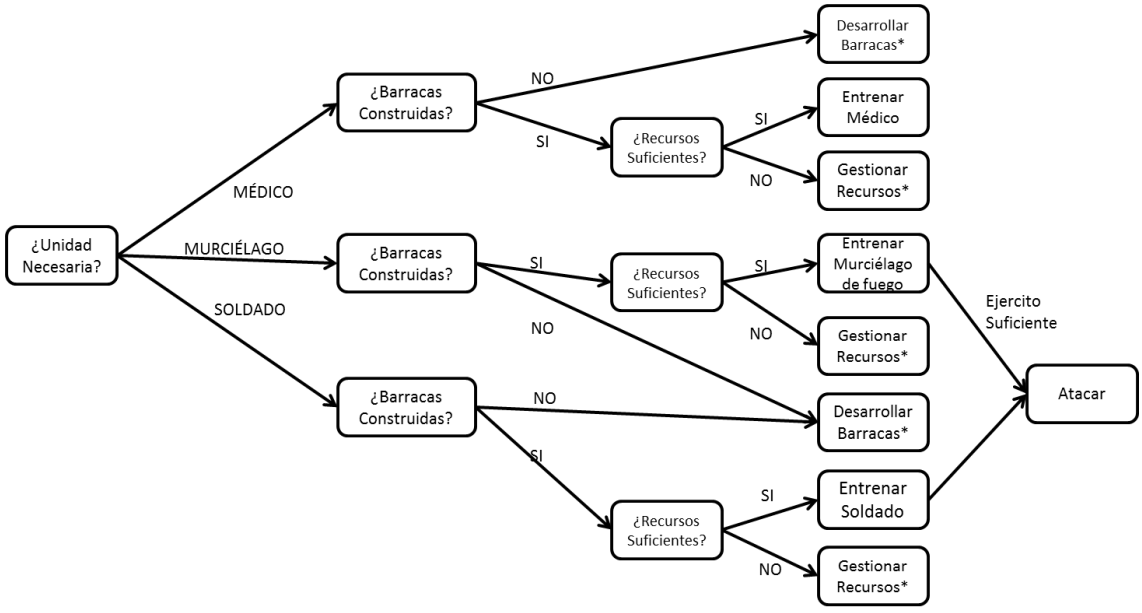

Figura 23. Diagrama del árbol del estado "Entrenar Unidad"

La figura 24 representa el árbol que desarrolla el estado Fabricar Vehículo. Como otros árboles vistos anteriormente, este árbol transita a otro estado en caso de requerirse más recursos o la construcción de un nuevo edificio. Como las unidades, los vehículos también forman parte de batallones.

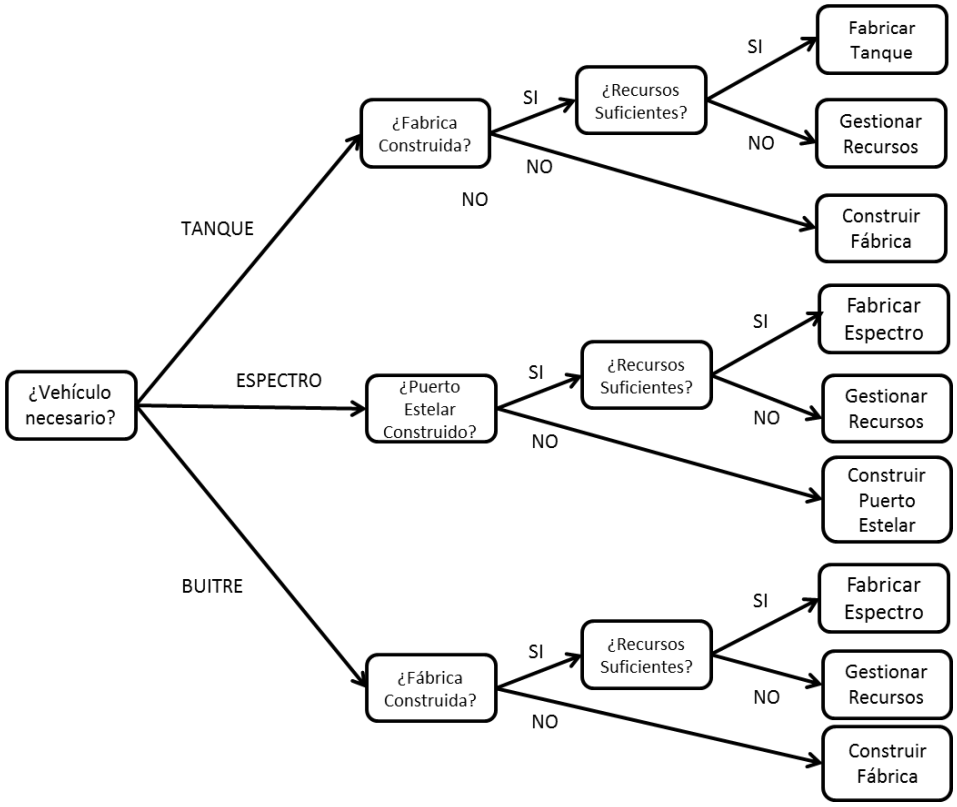

Figura 24. Diagrama del árbol del estado "Fabricar Vehículo"

Para construir edificios el árbol identifica el edificio que se desea construir y si se dispone de los recursos u otros requisitos necesarios. En caso de no ser así se transita al estado correspondiente (figura 25).

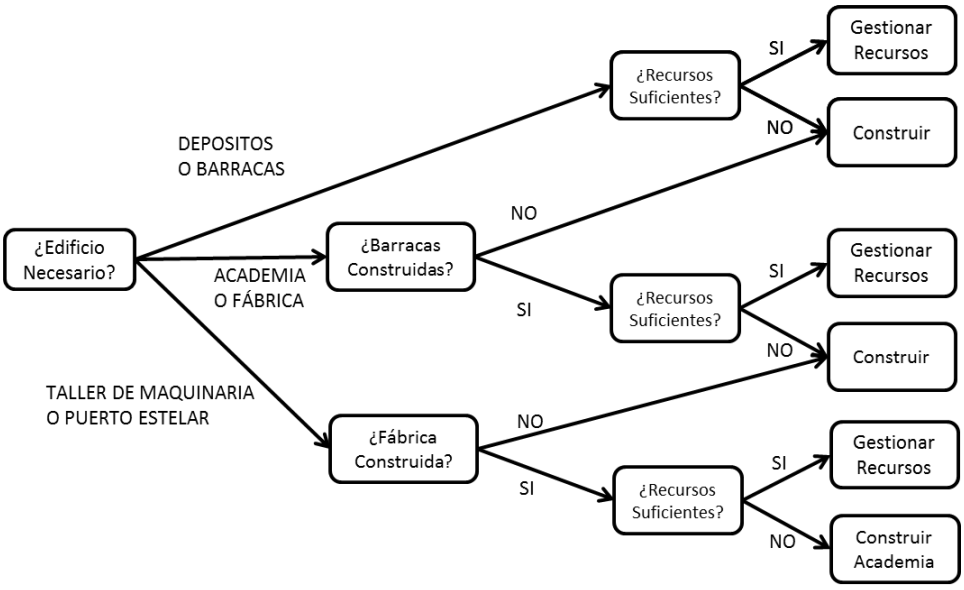

Figura 25. Diagrama del árbol del estado "Construir Edificios"

El estado atacar se representa en la figura 26. Este árbol de decisión decide la importancia del enemigo a atacar y si se dispone de batallones con las unidades suficientes para realizar el ataque.

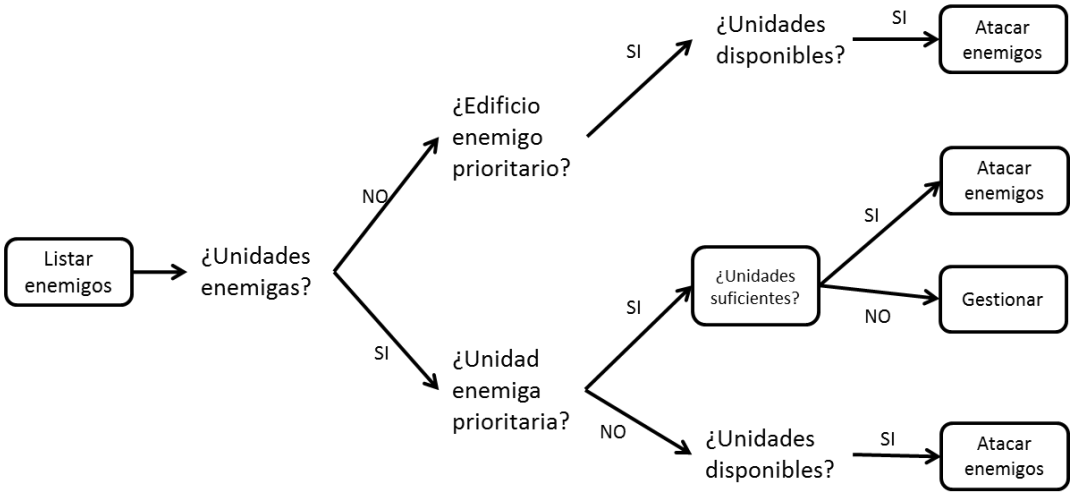

Figura 26. Diagrama del árbol del estado "Atacar"

La figura 27 detalla el estado defender, el cual analiza según la importancia de la unidad o edificio atacado y si merece la pena enviar más unidades a su defensa.

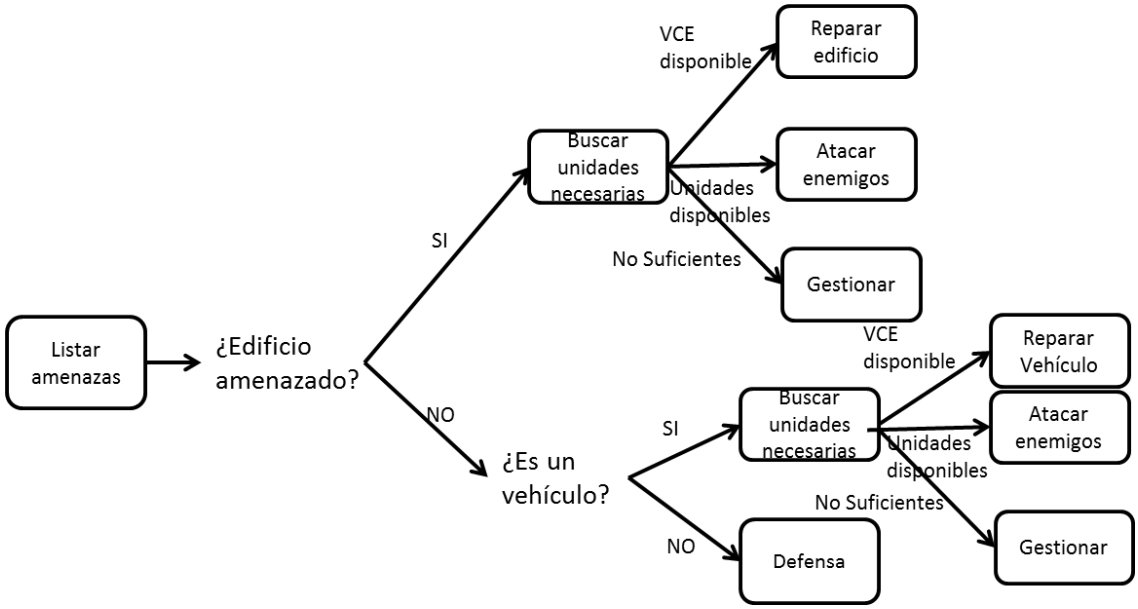

Figura 27. Diagrama del árbol del estado "Defender"

Por último, el estado Investigar (figura 28) se encarga de desarrollar nuevas tecnologías cuando se cumplen las condiciones necesarias.

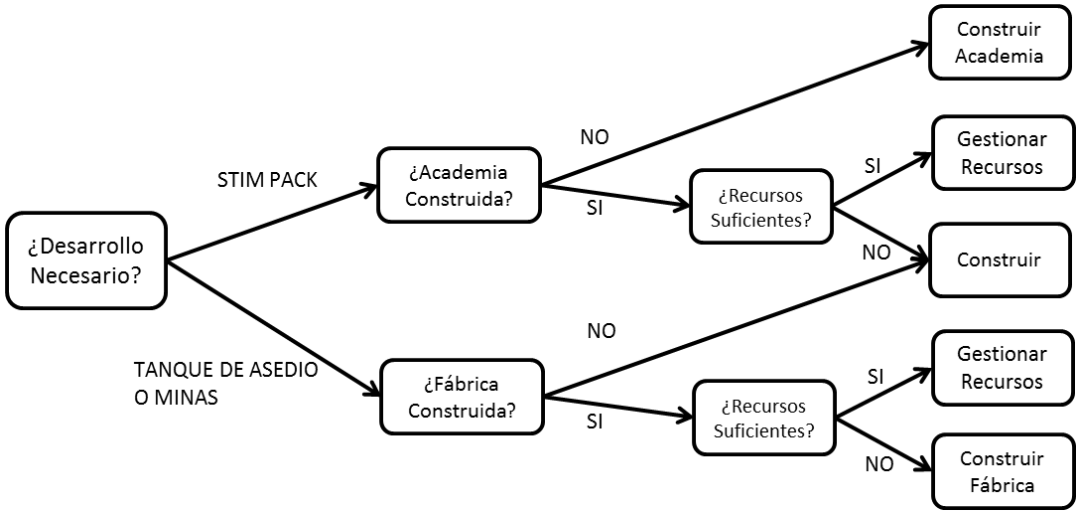

Figura 28. Diagrama del árbol del estado "Investigar"

3.3.2 Descripción general del sistema

El sistema (figura 29) implementa un jugador automático para el control del videojuego *StarCraft: Broodwar* utilizando BWAPI, un *framework* con el que poder interactuar con el juego. Para desarrollar el jugador automático se diseña una máquina de estados combinada con árboles de decisión donde implementar la IA del jugador con el objetivo de lograr vencer a los oponentes del juego.

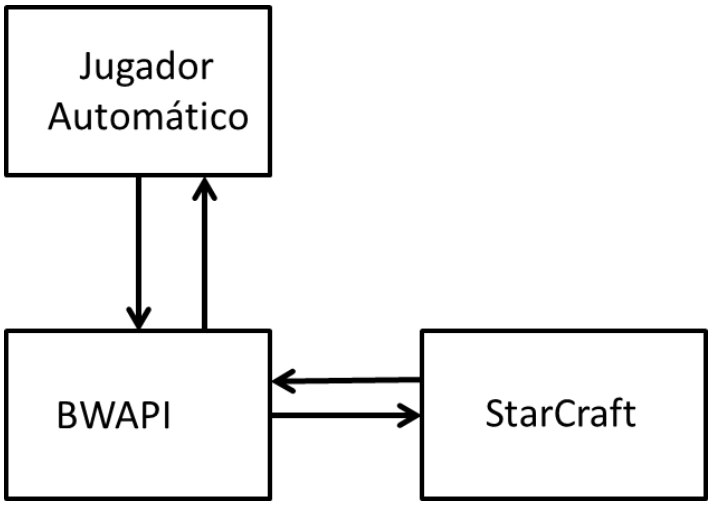

Figura 29. Diagrama de la estructura del sistema

3.3.3 Descripción de componentes

Seguidamente se realiza una descripción detallada de los componentes del sistema:

- StarCraft: Videojuego de estrategia en tiempo real sobre el que se realiza este proyecto. El sistema interactúa con el juego mediante un autómata para lo que necesita el interfaz BWAPI.

- BWAPI: Entorno de código abierto para interactuar con el videojuego StarCraft: Broodwar para poder crear robots que controlen el juego empleando la inteligencia artificial.

- Autómata: Sistema de inteligencia artificial creado para controlar el juego *StarCraft* basado en una combinación de una máquina de estado en la que cada uno de los estados se desarrolla mediante un árbol de decisión. Para realizarlo se han agrupado acciones del mismo tipo (por ejemplo entrenar soldado o entrenar murciélago de fuego) dentro del mismo estado, al cual determina si la acción a realizar es viable o requiere de otras acciones para lo cual transitará al estado correspondiente.

Capítulo 4: Experimentación

En este capítulo se presenta la experimentación realizada para comprobar el correcto funcionamiento del sistema. Para este proyecto se ha optado por la experimentación directa ya que el funcionamiento del sistema puede ser evaluado mediante una serie de experimentos sin la necesidad de un usuario.

4.1 Entorno de Pruebas

Las realizadas se han efectuado en un ordenador portátil con sistema operativo Windows 7, procesador Intel Core i3 a 2,27 Ghz, 4 GB de memoria RAM y tarjeta gráfica ATI Radeon con 512MB de memoria dedicada. Tanto StarCraft como Eclipse, BWAPI y Chaoslauncher se han instalado correctamente.

4.2 Pruebas unitarias

A continuación se describen las diferentes pruebas unitarias que han sido realizadas para comprobar el funcionamiento del jugador.

Recogida de minerales	
Prueba	01
Nombre	Recogida de Minerales
Descripción	Se cambian los parámetros que calculan el número de VCEs que deben de recolectar mineral o gas y se ejecuta el autómata a través de Chaoslauncher.
Objetivo	Comprobar que los VCEs con los que cuenta el autómata se reparten correctamente entre recolectar minerales o gas vespeno.
Resultado	El jugador distribuye los VCEs disponibles entre ambas tareas correctamente.
Problemas Encontrados	En ocasiones no se dispone de los VCEs mínimos para poder distribuirlos entre ambas tareas o la refinería de gas no llega a construirse.

Tabla 27 – Prueba de recogida de minerales

Entrenar médico	
Prueba	02
Nombre	Entrenar médico
Descripción	Observar si se completan los batallones de soldados con médicos cuando se disponen de suficientes soldados o murciélagos de fuego y se ha construido una academia.
Objetivo	Comprobar que se ejecuta la orden de entrenar médicos correctamente completando de esta manera cada batallón con al menos un médico.
Resultado	El autómata entrena médicos cuando tiene suficientes soldados para completar un batallón.
Problemas Encontrados	Hay que llegar a un mínimo de soldados entrenados que podrían tener que utilizarse en labores de defensa antes de poder completar el batallón.

Tabla 28 – Prueba de entrenar médico

Entrenar murciélago de fuego	
Prueba	03
Nombre	Entrenar murciélago de fuego
Descripción	Observar si se entrenan murciélagos de fuego cuando se dispone de academia.
Objetivo	Comprobar que se ejecuta la orden de entrenar murciélagos de fuego para disponer así de unidades con mayor poder de ataque.
Resultado	El autómata entrena murciélagos de fuego cuando se termina de construir la academia y se tienen los recursos suficientes.
Problemas Encontrados	No se encontraron problemas

Tabla 29 – Prueba de entrenar murciélago de fuego

Atacar enemigos	
Prueba	04
Nombre	Atacar enemigos
Descripción	Observar si el autómata ordena atacar enemigos cuando se han completado batallones con las unidades necesarias. Para observar su correcta ejecución se prueba a variar el número de unidades de los batallones o el número de estos que deben defender la base.
Objetivo	Comprobar que el autómata realiza la acción de atacar en el momento oportuno.
Resultado	El autómata ordena a cada unidad de los batallones disponibles (no realizando labores de defensa) la acción de atacar al enemigo.
Problemas Encontrados	No se encontraron problemas

Tabla 30 – Prueba de atacar enemigos

Defender	
Prueba	05
Nombre	Defender
Descripción	Observar si el autómata ordena desplazarse a las unidades y vehículos hasta la posición del edificio/unidad atacado.
Objetivo	Comprobar que el autómata detecta ataques del enemigo y responde correctamente.
Resultado	El autómata detecta el ataque y se ordena el desplazamiento de las unidades
Problemas Encontrados	Sin problemas encontrados

Tabla 31 – Prueba de defender

Construir Refinería	
Prueba	06
Nombre	Construir refinería
Descripción	El sistema es capaz de encontrar un geiser de gas vespeno y construir una refinería en él.
Objetivo	Comprobar que el sistema construye correctamente refinerías de gas vespeno.
Resultado	El sistema construye refinerías sobre el géiser más cercano a la base.
Problemas Encontrados	En ocasiones se ordena a más de un VCE la construcción de la refinería por lo que se "molestan" y no inician la construcción. El problema se soluciona al volver a realizar la orden de construcción.

Tabla 32 – Prueba de construir refinería

Entrenar VCE	
Prueba	07
Nombre	Entrenar VCE
Descripción	Cambio en los parámetros del sistema que determinan cuándo hay que entrenar más VCEs.
Objetivo	Comprobar que el sistema entrena el número de VCEs adecuados al tamaño del ejército.
Resultado	El sistema entrena los VCEs especificados en los parámetros
Problemas Encontrados	Hay que tener en cuenta que la partida comienza con cuatro VCEs ya disponibles.

Tabla 33 – Prueba de Entrenar VCE

Asignar batallón	
Prueba	08
Nombre	Asignar batallón
Descripción	Observar mediante mensajes si las unidades creadas se asignan al batallón adecuado.
Objetivo	Comprobar que el sistema está gestionando los batallones adecuadamente.
Resultado	El sistema gestiona bien los batallones.
Problemas Encontrados	Dependiendo de la evolución de la partida las unidades creadas pueden no ser suficientes para la creación de distintos batallones por lo que la prueba debe repetirse varias veces para comprobar el correcto funcionamiento.

Tabla 34 – Prueba de Asignar Batallón

Unidad destruida	
Prueba	09
Nombre	Unidad destruida
Descripción	Observar mediante mensajes si el autómata identifica las unidades destruidas por el enemigo.
Objetivo	Comprobar que el sistema está identificando las unidades perdidas.
Resultado	El sistema gestiona correctamente las pérdidas.
Problemas Encontrados	Fue necesaria la creación de un listado de las unidades creadas con su estado ya que al eliminarse una unidad no es posible recuperar más información de ella.

Tabla 35 – Prueba de Unidad destruida

Unidad completada	
Prueba	10
Nombre	Unidad completada
Descripción	Observar si el autómata recoge la información sobre las unidades creadas y su tipología mediante mensajes.
Objetivo	Comprobar que el sistema está identificando las unidades creadas correctamente.
Resultado	El sistema gestiona correctamente las unidades creadas.
Problemas Encontrados	Sin problemas encontrados.

Tabla 36 – Prueba de Unidad completada

A continuación se presentan los resultados de la pruebas del jugador automático de este proyecto en partidas completas. Se ha tomado como muestra 15 partidas diferentes, en 3 mapas distintos y jugando contra distintas razas:

Mapa Astral Balance					
Raza enemiga	Zerg	Terran	Protoss	Zerg	Terran
Tiempo de juego	8:49	6:37	6:02	9:32	6:10
Resultado de la partida	Derrota	Derrota	Derrota	Derrota	Victoria
Número de edificios construidos	8	8	7	11	6
Número de unidades entrenadas	46	39	33	51	40
Cantidad de mineral recogido	4.370	3.018	2.858	5.210	3.266
Cantidad de gas vespeno recogido	1.544	400	240	1.112	544
Número de centros de mando construidos	0	0	0	0	0
Número de refinerías de gas vespeno	1	1	1	1	1
Número de transiciones ejecutadas	16.754	12.795	11.969	18.975	12.248

Tabla 37 – Conjunto de pruebas con el mapa Astral Balance

Mapa Baby Steps					
Raza enemiga	Terran	Terran	Zerg	Protoss	Terran
Tiempo de juego	9:03	6:22	12:09	7:31	6:58
Resultado de la partida	Derrota	Victoria	Derrota	Derrota	Derrota
Número de edificios construidos	8	6	13	9	9
Número de unidades entrenadas	44	39	66	39	35
Cantidad de mineral recogido	2.938	2.674	6.202	2.834	2.666
Cantidad de gas vespeno recogido	744	360	1.560	752	848
Número de centros de mando construidos	0	0	0	0	0
Número de refinerías de gas vespeno	1	1	1	1	1
Número de transiciones ejecutadas	17.869	12.703	24.265	14.985	13.874

Tabla 38 – Conjunto de pruebas con el mapa Baby Steps

Mapa Binary Burghs					
Raza enemiga	Protoss	Terran	Terran	Zerg	Protoss
Tiempo de juego	7:39	6:34	6:58	9:37	6:02
Resultado de la partida	Derrota	Derrota	Derrota	Derrota	Derrota
Número de edificios construidos	6	6	7	9	6
Número de unidades entrenadas	33	35	36	51	38
Cantidad de mineral recogido	2.378	2.858	2.594	4.530	2626
Cantidad de gas vespeno recogido	272	304	584	1.288	240
Número de centros de mando construidos	0	0	0	0	0
Número de refinerías de gas vespeno	1	1	1	1	1
Número de transiciones ejecutadas	14.955	13.063	13.844	19.130	11876

Tabla 39 – Conjunto de pruebas con el mapa Binary Burghs

Como resultado de las pruebas realizadas se observa que el jugador automático realiza correctamente las funciones de gestión de recursos, creación de unidades y gestión de edificios pero la estrategia implementada no consigue que el jugador sea del todo eficaz a la hora de luchar contra los enemigos ya que solo gana un pequeño porcentaje de las partidas que juega (13%). Con más análisis se observa que el mejor desempeño del jugador automático se obtiene contra la raza Terran y en partidas que se resuelven en un periodo corto de tiempo, cuando el enemigo no ha desarrollado todo su ejército.

Capítulo 5: Gestión del proyecto

Durante la realización de este proyecto se ha seguido un modelo de desarrollo en espiral. Propuesto originalmente por Barry Boehm en 1986, es un modelo de proceso de software evolutivo que conjuga la naturaleza iterativa de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial. Se comienza produciendo una pequeña parte del sistema (completamente funcional) y una vez completada, se procede a crear una segunda parte, acoplada a la primera, de manera de que en cada iteración, se obtiene una versión aumentada del sistema hasta acabar. El modelo en espiral se divide para cada ejecución del desarrollo en cuatro pasos principales:

1. Determinar o fijar los objetivos. En este paso se definen los objetivos específicos de la iteración para posteriormente identificar las limitaciones del proceso y del sistema de software. Además se diseña una planificación detallada de gestión y se identifican los riesgos y sus alternativas.
2. Análisis del riesgo. En este paso se efectúa un análisis detallado de cada uno de los riesgos identificados del proyecto, se definen los pasos a seguir para reducir los riesgos y después de su análisis se planean estrategias alternativas.
3. Desarrollar, verificar y validar. En este tercer paso, después del análisis de riesgo, se elige un paradigma para el desarrollo del sistema de software y se implementa posteriormente.
4. Planificar. En este último paso es donde el proyecto se revisa y se toma la decisión si se debe continuar con un ciclo posterior al de la espiral. Si se decide continuar, se desarrollan los planes para la siguiente fase del proyecto.

En la figura 30 se observan las distintas fases en las que se divide el modelo de desarrollo en espiral.

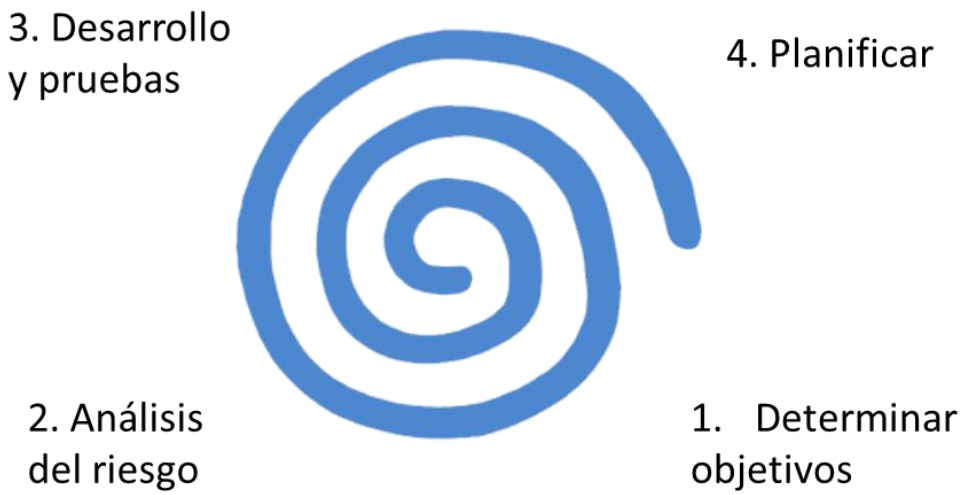

Figura 30. Diagrama del modelo de desarrollo en espiral

En el modelo de desarrollo en espiral las tareas se adaptan a las características del proyecto que va a emprenderse. Para proyectos pequeños, el número de tareas de trabajo y su formalidad es bajo. Para proyectos mayores y más críticos cada región de tareas contiene tareas de trabajo que se definen para lograr un nivel más alto de formalidad. En todos los casos, se aplican las actividades de protección (por ejemplo: gestión de configuración del software y garantía de calidad del software).

Cuando empieza este proceso evolutivo, el equipo de ingeniería del software gira alrededor de la espiral en la dirección de las agujas del reloj, comenzando por el centro. El primer circuito de la espiral puede producir el desarrollo de una especificación de productos; los pasos siguientes en la espiral se podrían utilizar para desarrollar un prototipo y, progresivamente, versiones más sofisticadas del software. Cada paso por la región de planificación produce ajustes en el plan del proyecto.

El coste y la planificación se ajustan con la realimentación ante la evaluación del cliente. Además, el gestor del proyecto ajusta el número planificado de iteraciones requeridas para completar el software.

El modelo en espiral es un enfoque realista del desarrollo de sistemas y de software a gran escala. Como el software evoluciona, a medida que progresa el proceso el desarrollador y el cliente comprenden y reaccionan mejor ante riesgos en cada uno de los niveles evolutivos.

El modelo en espiral demanda una consideración directa de los riesgos técnicos en todas las etapas del proyecto, y, si se aplica adecuadamente, debe reducir los riesgos antes de que se conviertan en problemáticos. Pero al igual que otros paradigmas, el modelo en espiral no es la panacea. Puede resultar difícil convencer a grandes clientes (particularmente en situaciones bajo contrato) de que el enfoque evolutivo es controlable.

Requiere una considerable habilidad para la evaluación del riesgo, y cuenta con esta habilidad para el éxito. Si un riesgo importante no es descubierto y gestionado, indudablemente surgirán problemas. Finalmente, el modelo no se ha utilizado tanto como los paradigmas lineales secuenciales o de construcción de prototipos. Todavía tendrán que pasar muchos años antes de que se determine con absoluta certeza la eficacia de este nuevo e importante paradigma.

Las principales ventajas de este modelo son:

- No requiere una definición completa de los requisitos del software a desarrollar para comenzar su desarrollo ya que puede adaptarse a lo largo de la vida del software.
- Se monitorizan y controlan los riegos tempranamente por lo que es más fácil poder corregirlos a tiempo.

Las desventajas a destacar en este tipo de desarrollos son las siguientes:

- Es complicado evaluar los riesgos.
- Se requiere la participación continua por parte del cliente.
- Supone una carga de trabajo adicional al volver producir inicialmente una especificación completa de los requisitos cuando se modifica o mejora el software.

5.1 Descripción de las fases del proyecto

Las fases de las que se compone el proyecto son las siguientes:

- Fase 1: La primera fase tuvo como objetivo principal la familiarización del autor con el videojuego StarCraft así como con el entorno BWAPI para poder desarrollar jugadores automáticos.
- Fase 2: Una vez familiarizado con los conceptos básicos de BWAPI, en la siguiente fase se definió el autómata a desarrollar en el proyecto. Para ello se diseñó una máquina de estados que recogiese las funciones a realizar por el autómata y árboles de decisión para cada nodo de la máquina donde se desarrollan las acciones asociadas a cada estado. Además de la definición de la máquina de estados, se avanza en la implementación de acciones básicas como construcción de edificios y entrenamiento de unidades.
- Fase 3: En esta fase se desarrolla el cliente con todas las acciones de la máquina de estados y se ajustan sus parámetros para comprobar que se cumple el objetivo del proyecto. En esta fase también se realiza esta memoria.

5.2 Planificación

En el diagrama de Gantt (figura 31) se recoge la planificación de todas las fases del proyecto:

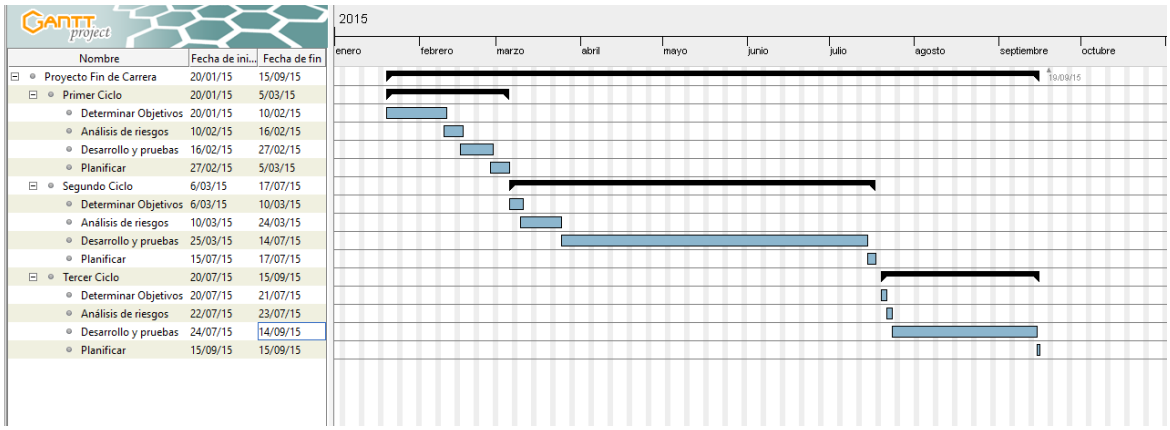

Figura 31. Diagrama de Gantt del proyecto

5.3 Presupuesto

A continuación se presenta un presupuesto detallado del coste de desarrollo de este proyecto.

- Coste de personal:

  Para el cálculo del coste de personal se han utilizado las tablas publicadas por el ministerio de economía en el BOE de 23 de marzo de 2015:

Rol	Nº Personas	Horas	Coste por horas	Coste Total
Programador	1	300,0	17,8	5.336,2
Analista	1	80,0	28,1	2.244,2
Jefe de Proyecto	1	60,0	30,6	1.836,2
Total				**9.416,6**

Tabla 40 – Costes de personal

- Coste de Material:

  Incluye el hardware y software utilizados para la realización del proyecto. Para determinar el periodo de amortización se ha utilizado el permitido en las tablas para el Impuesto de Sociedades de 2015 que establece un máximo de 6 años para sistemas y programas informáticos.

Nombre	Unidades	Coste Unitario	Vida útil (meses)	Periodo Amortizado (meses)	Coste Total
Ordenador	1	790,0	72	9	98,8
Office 2010	1	119,0	72	9	14,9
Conexión internet/mes	9	50,0	-	-	450,0
StarCraft: Broodwar	1	19,9	72	9	2,5
Eclipse	1	0,0	72	9	0,0
BWAPI	1	0,0	72	9	0,0
Chaos Launcher	1	0,0	72	9	0,0
Total					**566,1**

Tabla 41 – Costes de material

- Coste total:

  Para el cálculo del coste total de la realización del proyecto se han tenido en cuenta unos costes indirectos que representan el 20% del total.

Concepto	Coste
Coste de personal	9.416,6
Coste de material	566,1
Costes Indirectos	1.996,5
Total	**11.979,2**

Tabla 42 – Coste del proyecto

La cantidad total invertida en este proyecto asciende a once mil novecientos setenta y nueve Euros.

Capítulo 6: Conclusiones y trabajos futuros

En este apartado se incluyen las conclusiones obtenidas tras la realización del proyecto. En primer lugar se presentan las conclusiones generales sobre el proyecto realizado. A continuación se presentan las conclusiones obtenidas para cada uno de los objetivos que se incluyeron en el capítulo 1 del documento y se finaliza con los posibles trabajos futuros que podrían realizarse sobre este proyecto.

6.1 Conclusiones generales

El objetivo del proyecto, como se describe anteriormente, es el desarrollo de un autómata para el videojuego StarCraft: Broodwar a través de BWAPI. Este objetivo se ha realizado con éxito, ya que el autómata desarrollado es capaz de controlar un jugador de la raza Terran realizando la mayor parte de las acciones disponibles en el juego.

Una vez concluido el proyecto, se pueden obtener las siguientes conclusiones:

1. Estrategia: Uno de los puntos más importantes para tener éxito en el videojuego StarCraft es la estrategia a seguir. Las observaciones realizadas en las pruebas del proyecto sumadas a la experiencia anterior como jugador indican que una estrategia más agresiva tiene más posibilidades de éxito que una centrada en el desarrollo de un gran ejército. Por lo tanto, la conclusión es que la estrategia a seguir debe centrarse en desarrollar los vehículos y unidades necesarios para atacar con éxito al enemigo, dejando en un segundo lado la defensa de la base propia, aunque teniendo también este factor en cuenta ya que es imprescindible para poder seguir produciendo.

   Un jugador más agresivo obligará al contrario a defender en vez de poder enviar sus recursos a atacar las bases del jugador. Al atacar también se reducen los recursos del enemigo, por lo que sus posibilidades de un contraataque se reducen. A pesar de la importancia del ataque no hay que olvidar la importancia de tener cubiertas las necesidades de recolección de recursos, entrenamiento de nuevas unidades o fabricación de vehículos.

2. Velocidad: La velocidad en la toma de decisiones es vital para el éxito en StarCraft. Adelantarse al enemigo desarrollando un ejército suficiente para atacar y realizar el ataque es, probablemente, uno de los requisitos más importantes para vencer al enemigo.

La velocidad también es importante a la hora de recolectar recursos ya que es crucial llegar a las zonas de minerales o geiseres de vespeno antes que el enemigo.

3. Simplicidad: Para implementar el autómata la simplicidad de la estructura a desarrollar es un punto importante a la hora de realizar mejoras o cambios ya que permite ahorrar mucho tiempo de desarrollo y pruebas.

Este proyecto tiene en cuenta las conclusiones anteriores, si bien es posible desarrollarlas como se verá más adelante.

6.2 Conclusiones referentes a los objetivos

El objetivo principal del proyecto se descompone en los objetivos que se analizan a continuación:

1- Análisis del videojuego *StarCraft: Broodwar*:

Un primer paso del proyecto era familiarizarse con el videojuego, su mecánica de juego y las posibles estrategias a aplicar. Para realizar este objetivo fue necesario jugar al juego, primero en modo campaña para aprender las características de las distintas razas y unidades y después contra la máquina en partidas personalizadas con características similares a las que se emplean en las partidas con el jugador automático desarrollado. Además fue necesario investigar en distintas páginas web y foros sobre las ventajas e inconvenientes de las distintas unidades así como las distintas estrategias a emplear.

2- Definición del sistema de toma de decisiones:

Para la definición del autómata se parte del punto anterior, ya que es imprescindible tener un amplio conocimiento del juego para analizar las funciones que debe de realizar el jugador así como las condiciones con las que se debe de ejecutar cada acción. También es importante el estudio de las diferentes opciones para su implementación habiéndose escogido la combinación de máquina de estados y árbol de decisión por ser rápida, siendo la velocidad de proceso un punto a tener en cuenta en este proyecto, y de fácil implementación teniendo en cuenta las limitaciones de tiempo con las que se contaban.

3- Estudio de las técnicas más adecuadas para implementar IA en un videojuego de estrategia en tiempo real:

   Tras plantear varias alternativas se decidió implementar el jugador automático utilizando una combinación de máquina de estados y árboles de decisión por ser una de las opciones más sencillas de implementar sistemas complejos además de presentar ventajas en la velocidad de ejecución.

4- Análisis y estudio de la API:

   Para la realización del proyecto es imprescindible conocer la estructura de BWAPI, ya que es la interfaz con la que el autómata se comunica con el juego. Este objetivo se realizó con éxito a pesar de algunas dificultades principalmente debidas a la poca experiencia del autor con el lenguaje de programación Java.

5- Comprobación del correcto funcionamiento del jugador automático desarrollado:

   Se han realizado pruebas, tanto parciales como jugando partidas completas, para verificar que el jugador automático realiza las acciones requeridas tal y como se espera. Con los resultados obtenidos esta experimentación se observa que el jugador automático realiza correctamente las funciones implementadas aunque no tiene un desempeño óptimo frente a los adversarios ya que el porcentaje de partidas acabadas en victoria es bajo.

6.3 Trabajos futuros

Este proyecto crea un jugador automático capaz de jugar al videojuego *StarCraft: Broodwar* implementando un gran número de acciones. Sin embargo, a continuación se detallan posibles mejoras en el sistema:

- El autómata se está basado en un conocimiento total del mapa donde se desarrolla la partida por la posibilidad que permite BWAPI de disponer de toda la información. Para que las condiciones de funcionamiento del autómata fuesen las mismas que las de un jugador humano habría que rediseñar la estrategia implementada para tener en cuenta que solo se conocen ciertas zonas del mapa. Para ello habría que añadir acciones de exploración y recogida de datos para actuar posteriormente.

- El proyecto solo está desarrollado para jugar con la raza Terran. Sería muy interesante la posibilidad de añadir el control de las razas Protoss y Zerg.

- En un gran número de ocasiones el autómata es vencido por su oponente por lo que sería aconsejable la revisión del autómata para hacerlo más eficaz frente a los enemigos. Para esto se podrían revisar las condiciones de cada estado o implementar nuevas acciones.

Anexos

Manual de instalación

Los requisitos del sistema para instalar *StarCraft* son los siguientes:

- Sistema operativo: Windows XP o superior
- Procesador (CPU): 1,6 GHz
- Memoria RAM: 384 MB
- Tarjeta gráfica(GPU): Pantalla a 1024x768 píxeles de resolución
- Espacio en disco (HDD): 900 MB

Los pasos para instalar el entorno del proyecto son:

1. Instalación de *StarCraft: Broodwar*.
   Para ello es necesario tener instalado previamente *StarCraft*. Si no es así, inserte el disco de *StarCraft* y a continuación ejecute "Setup.exe" y siga las instrucciones por pantalla. Una vez terminada la instalación realice la mima operación con el disco de *Broodwar*.
2. Una vez instalado el juego compruebe que está actualizado a la última versión. Si en la esquina inferior derecha de la pantalla de inicio la versión es distinta a la 1.16.1 (ver figura 32) deberá actualizar el juego desde la página oficial de Blizzard.

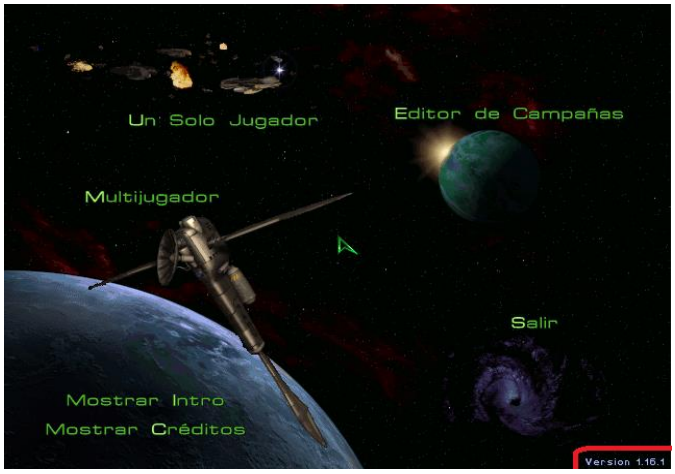

Figura 32. Pantalla de inicio de StarCraft

3. Instalar Visual C++.
4. Instalar Eclipse
5. Descargar BWAPI 3.7 y Chaoslauncher. Ambos están disponibles en http://code.google.com/p/bwapi/downloads/list.
6. Descomprimir el archivo descargado y ejecutar el script install.exe. Si no se ejecuta correctamente hay que seguir los siguientes pasos:
    i. Copiar el contenido de WINDOWS/ al directorio C:\WINDOWS
    ii. Copiar el contenido de Starcraft/ a la carpeta del videojuego Starcraft.
7. Abrir el fichero ExampleProjects.sln con Visual Studio y compilar el proyecto como RELEASE

Si se utiliza Windows 7 o superior y al ejecutar el jugador automático aparece en pantalla el mensaje "Failed to load AI module NULL" será necesario otorgar derechos de administrador tanto a Chaoslauncher como a Eclipse.

Manual de usuario

Para ejecutar el jugador automático hay que realizar la siguiente secuencia:

1. Descargar el código fuente del proyecto.
2. Abrir el proyecto “StarcraftIA” y ejecutar la clase “ClienteIA.java”
3. Abrir Chaoslauncher (figura 33)

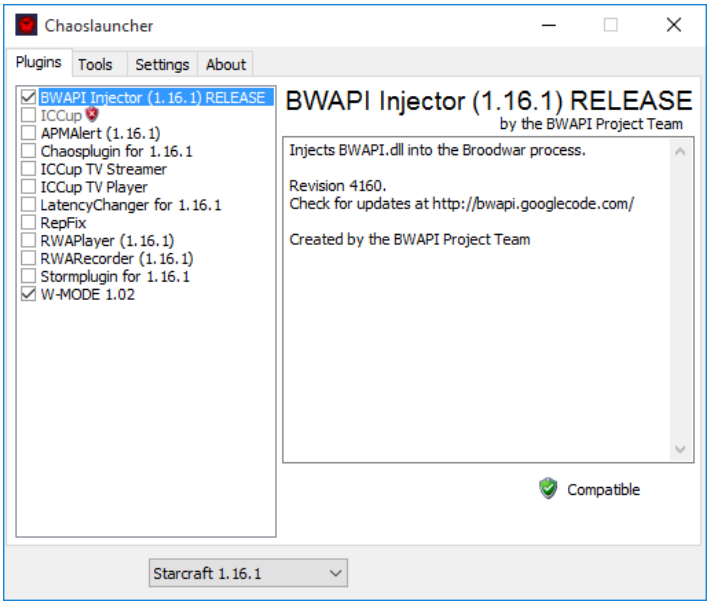

Figura 33. Imagen de Chaoslauncher

4. Pulsar botón “Start”
5. En la pantalla de selección de jugadores de *StarCraft* seleccionar Terran como raza del jugador.
6. La partida comenzará con el jugador automático tomando el control del juego.

Definiciones

API: Abreviatura de Interfaz de programación de aplicaciones (del inglés *Application Programming Interface*). Es el conjunto es de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

AEVI: Asociación Española de Videojuegos

Bot: Aféresis de robot

E3: Electronic Entertainment Expo. La convención de videojuegos más importante de la industria. Se celebra cada año en el Centro de Convenciones de Los Ángeles, California.

Referencias

[1] Artificial Intelligence and Soft Computing – ICAISC 2008

[2] Neal Ford. The Productive Programmer (2008) O'Reilly Media

[3] www.rae.es

[4] www.videogameconsolelibrary.com

[5] http://www.bloomberg.com/slideshow/2012-11-16/-pong-turns-40-but-it-s-not-the-oldest-video-game.html

[6] http://archive.computerhistory.org/projects/chess/related_materials/text/2-0%20and%202-1.Programming_a_computer_for_playing_chess.shannon/2-0%20and%202-1.Programming_a_computer_for_playing_chess.shannon.062303002.pdf

[7] http://www.otakufreaks.com/historia-de-los-videojuegos-el-origen-y-los-inicios/

[8] http://www.douglashistory.co.uk/history/sandy_douglas.htm

[9] http://www.computerhistory.org/pdp-1/spacewar/

[10] http://infolab.stanford.edu/pub/voy/museum/galaxy.html

[11] Ralph Baer. Videogames: In the Beginning (2005) Rolenta Press

[12] http://www.pong-story.com/odyssey.htm

[13] http://www.adventuregamers.com/articles/view/17880

[14] http://www.hobbyconsolas.com/noticias/gta-v-supera-800-millones-dolares-su-primer-dia-57086

[15] http://www.1up.com/features/essential-50-herzog-zwei

[16] http://strategywiki.org/wiki/Dune_II:_The_Building_of_a_Dynasty

[17] http://gamasutra.com/view/feature/181339/the_making_of_warcrafts_.php?print=1

[18]
http://web.archive.org/web/20080118005127/http://www.microsoft.com/games/empires/behind_dave.htm

[19] http://uk.ign.com/articles/1999/02/09/lessons-on-the-art-of-war

[20] http://features.slashdot.org/story/08/10/15/1639237/blizzard-answers-your-questions-from-blizzcon

[21] http://www.guinnessworldrecords.com/world-records/best-selling-strategy-game-for-pc

[22] Alan Turing: The Enigma (1992) Burnett Books Ltd

[23] The American Heritage Dictionary del Idioma Inglés, Cuarta Edición (Houghton Mifflin Company)

[24] http://elpais.com/diario/2011/10/27/necrologicas/1319666402_850215.html

[25] https://www.etsisi.upm.es/www.eui.upm.es/museo_virtual/2g/inteligenciaartificial

[26] Ian Millington y John Funge, Artificial Intelligence for Games (2009) Ed. CRC Press

[27] http://aigamedev.com/open/editorial/2014-awards/

[28] http://idelab.uva.es/algoritmo

[29] Maria Isabel Alfonso Galipienso, Inteligencia artificial (2003) Ed. Paraninfo

[30] https://software.intel.com/en-us/articles/designing-artificial-intelligence-for-games-part-1